

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Application Program Interface for Network Software  
Platform**

Inventor(s):  
Brad Abrams

**E1792821794**

ATTORNEY'S DOCKET NO. MS1-864US

1 **TECHNICAL FIELD**

2 This invention relates to network software, such as Web applications, and to  
3 computer software development of such network software. More particularly, this  
4 invention relates to an application program interface (API) that facilitates use of a  
5 network software platform by application programs and computer hardware.

6  
7 **BACKGROUND**

8 Very early on, computer software came to be categorized as "operating  
9 system" software or "application" software. Broadly speaking, an application is  
10 software meant to perform a specific task for the computer user such as solving a  
11 mathematical equation or supporting word processing. The operating system is  
12 the software that manages and controls the computer hardware. The goal of the  
13 operating system is to make the computer resources available to the application  
14 programmer while at the same time, hiding the complexity necessary to actually  
15 control the hardware.

16 The operating system makes the resources available via functions that are  
17 collectively known as the Application Program Interface or API. The term API is  
18 also used in reference to a single one of these functions. The functions are often  
19 grouped in terms of what resource or service they provide to the application  
20 programmer. Application software requests resources by calling individual API  
21 functions. API functions also serve as the means by which messages and  
22 information provided by the operating system are relayed back to the application  
23 software.

24 In addition to changes in hardware, another factor driving the evolution of  
25 operating system software has been the desire to simplify and speed application

1 software development. Application software development can be a daunting task,  
2 sometimes requiring years of developer time to create a sophisticated program  
3 with millions of lines of code. For a popular operating system such as Microsoft  
4 Windows®, application software developers write thousands of different  
5 applications each year that utilize the operating system. A coherent and usable  
6 operating system base is required to support so many diverse application  
7 developers.

8 Often, development of application software can be made simpler by making  
9 the operating system more complex. That is, if a function may be useful to several  
10 different application programs, it may be better to write it once for inclusion in the  
11 operating system, than requiring dozens of software developers to write it dozens  
12 of times for inclusion in dozens of different applications. In this manner, if the  
13 operating system supports a wide range of common functionality required by a  
14 number of applications, significant savings in applications software development  
15 costs and time can be achieved.

16 Regardless of where the line between operating system and application  
17 software is drawn, it is clear that for a useful operating system, the API between  
18 the operating system and the computer hardware and application software is as  
19 important as efficient internal operation of the operating system itself.

20 Over the past few years, the universal adoption of the Internet, and  
21 networking technology in general, has changed the landscape for computer  
22 software developers. Traditionally, software developers focused on single-site  
23 software applications for standalone desktop computers, or LAN-based computers  
24 that were connected to a limited number of other computers via a local area  
25 network (LAN). Such software applications were typically referred to as “shrink

1 wrapped" products because the software was marketed and sold in a shrink-  
2 wrapped package. The applications utilized well-defined APIs to access the  
3 underlying operating system of the computer.

4 As the Internet evolved and gained widespread acceptance, the industry  
5 began to recognize the power of hosting applications at various sites on the World  
6 Wide Web (or simply the "Web"). In the networked world, clients from anywhere  
7 could submit requests to server-based applications hosted at diverse locations and  
8 receive responses back in fractions of a second. These Web applications, however,  
9 were typically developed using the same operating system platform that was  
10 originally developed for standalone computing machines or locally networked  
11 computers. Unfortunately, in some instances, these applications do not adequately  
12 transfer to the distributed computing regime. The underlying platform was simply  
13 not constructed with the idea of supporting limitless numbers of interconnected  
14 computers.

15 To accommodate the shift to the distributed computing environment being  
16 ushered in by the Internet, Microsoft Corporation is developing a network  
17 software platform known as the ".NET" platform (read as "Dot Net"). The  
18 platform allows developers to create Web services that will execute over the  
19 Internet. Such a dynamic shift requires a new ground-up design of an entirely new  
20 API.

21 In response to this challenge, the inventors developed a unique set of API  
22 functions for Microsoft's .NET™ platform.  
23  
24  
25

## **SUMMARY**

An application program interface (API) provides a set of functions for application developers who build Web applications on a network platform, such as Microsoft Corporation's .NET™ platform.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

The same numbers are used throughout the drawings to reference like features.

Fig. 1 illustrates a network architecture in which clients access Web services over the Internet using conventional protocols.

Fig. 2 is a block diagram of a software architecture for Microsoft's .NET™ platform, which includes an application program interface (API).

Fig. 3 is a block diagram of unique namespaces supported by the API, as well as function classes of the various API functions.

Fig. 4 is a block diagram of an exemplary computer that may execute all or part of the software architecture.

## **BRIEF DESCRIPTION OF ACCOMPANYING COMPACT DISC**

Accompanying this specification is a compact disc that stores a compiled HTML help file identifying the API (application program interface) for Microsoft's .NET™ network platform. The file is named "cpref.chm" and was created on June 8, 2001. It is 30.81 Mbytes in size. The file can be executed on a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98,

Windows® 2000, etc.). The compiled HTML help file stored on the compact disk is hereby incorporated by reference.

Additionally, the APIs contained in the compiled HTML help file are also provided in approximately 100 separate text files named "NamespaceName.txt". The text files comply with the ASCII format.

The compact disc itself is a CD-ROM, and conforms to the ISO 9660 standard.

### **DETAILED DESCRIPTION**

This disclosure addresses an application program interface (API) for a network platform upon which developers can build Web applications and services. More particularly, an exemplary API is described for the .NET™ platform created by Microsoft Corporation. The .NET™ platform is a software platform for Web services and Web applications implemented in the distributed computing environment. It represents the next generation of Internet computing, using open communication standards to communicate among loosely coupled Web services that are collaborating to perform a particular task.

In the described implementation, the .NET™ platform utilizes XML (extensible markup language), an open standard for describing data. XML is managed by the World Wide Web Consortium (W3C). XML is used for defining data elements on a Web page and business-to-business documents. XML uses a similar tag structure as HTML; however, whereas HTML defines how elements are displayed, XML defines what those elements contain. HTML uses predefined tags, but XML allows tags to be defined by the developer of the page. Thus, virtually any data items can be identified, allowing Web pages to function like

1 database records. Through the use of XML and other open protocols, such as  
2 Simple Object Access Protocol (SOAP), the .NET™ platform allows integration of  
3 a wide range of services that can be tailored to the needs of the user. Although the  
4 embodiments described herein are described in conjunction with XML and other  
5 open standards, such are not required for the operation of the claimed invention.  
6 Other equally viable technologies will suffice to implement the inventions  
7 described herein.

8 As used herein, the phrase application program interface or API includes  
9 traditional interfaces that employ method or function calls, as well as remote calls  
10 (e.g., a proxy, stub relationship) and SOAP/XML invocations.

#### 11 12 EXEMPLARY NETWORK ENVIRONMENT

13 Fig. 1 shows a network environment 100 in which a network platform, such  
14 as the .NET™ platform, may be implemented. The network environment 100  
15 includes representative Web services 102(1), ..., 102(N), which provide services  
16 that can be accessed over a network 104 (e.g., Internet). The Web services,  
17 referenced generally as number 102, are programmable application components  
18 that are reusable and interact programmatically over the network 104, typically  
19 through industry standard Web protocols, such as XML, SOAP, WAP (wireless  
20 application protocol), HTTP (hypertext transport protocol), and SMTP (simple  
21 mail transfer protocol) although other means of interacting with the Web services  
22 over the network may also be used, such as Remote Procedure Call (RPC) or  
23 object broker type technology. A Web service can be self-describing and is often  
24 defined in terms of formats and ordering of messages.

Web services 102 are accessible directly by other services (as represented by communication link 106) or a software application, such as Web application 110 (as represented by communication links 112 and 114). Each Web service 102 is illustrated as including one or more servers that execute software to handle requests for particular services. Such services often maintain databases that store information to be served back to requesters. Web services may be configured to perform any one of a variety of different services. Examples of Web services include login verification, notification, database storage, stock quoting, location directories, mapping, music, electronic wallet, calendar/scheduler, telephone listings, news and information, games, ticketing, and so on. The Web services can be combined with each other and with other applications to build intelligent interactive experiences.

The network environment 100 also includes representative client devices 120(1), 120(2), 120(3), 120(4), ..., 120(M) that utilize the Web services 102 (as represented by communication link 122) and/or the Web application 110 (as represented by communication links 124, 126, and 128). The clients may communicate with one another using standard protocols as well, as represented by an exemplary XML link 130 between clients 120(3) and 120(4).

The client devices, referenced generally as number 120, can be implemented many different ways. Examples of possible client implementations include, without limitation, portable computers, stationary computers, tablet PCs, televisions/set-top boxes, wireless communication devices, personal digital assistants, gaming consoles, printers, photocopiers, and other smart devices.

The Web application 110 is an application designed to run on the network platform and may utilize the Web services 102 when handling and servicing

requests from clients 120. The Web application 110 is composed of one or more software applications 130 that run atop a programming framework 132, which are executing on one or more servers 134 or other computer systems. Note that a portion of Web application 110 may actually reside on one or more of clients 120. Alternatively, Web application 110 may coordinate with other software on clients 120 to actually accomplish its tasks.

The programming framework 132 is the structure that supports the applications and services developed by application developers. It permits multi-language development and seamless integration by supporting multiple languages. It supports open protocols, such as SOAP, and encapsulates the underlying operating system and object model services. The framework provides a robust and secure execution environment for the multiple programming languages and offers secure, integrated class libraries.

The framework 132 is a multi-tiered architecture that includes an application program interface (API) layer 142, a common language runtime (CLR) layer 144, and an operating system/services layer 146. This layered architecture allows updates and modifications to various layers without impacting other portions of the framework. A common language specification (CLS) 140 allows designers of various languages to write code that is able to access underlying library functionality. The specification 140 functions as a contract between language designers and library designers that can be used to promote language interoperability. By adhering to the CLS, libraries written in one language can be directly accessible to code modules written in other languages to achieve seamless integration between code modules written in one language and code modules written in another language. One exemplary detailed implementation of a CLS is

described in an ECMA standard created by participants in ECMA TC39/TG3.

The reader is directed to the ECMA web site at [www.ecma.ch](http://www.ecma.ch).

The API layer 142 presents groups of functions that the applications 130 can call to access the resources and services provided by layer 146. By exposing the API functions for a network platform, application developers can create Web applications for distributed computing systems that make full use of the network resources and other Web services, without needing to understand the complex interworkings of how those network resources actually operate or are made available. Moreover, the Web applications can be written in any number of programming languages, and translated into an intermediate language supported by the common language runtime 144 and included as part of the common language specification 140. . In this way, the API layer 142 can provide methods for a wide and diverse variety of applications.

Additionally, the framework 132 can be configured to support API calls placed by remote applications executing remotely from the servers 134 that host the framework. Representative applications 148(1) and 148(2) residing on clients 120(3) and 120(M), respectively, can use the API functions by making calls directly, or indirectly, to the API layer 142 over the network 104.

The framework may also be implemented at the clients. Client 120(3) represents the situation where a framework 150 is implemented at the client. This framework may be identical to server-based framework 132, or modified for client purposes. Alternatively, the client-based framework may be condensed in the event that the client is a limited or dedicated function device, such as a cellular phone, personal digital assistant, handheld computer, or other communication/computing device.

## DEVELOPERS' PROGRAMMING FRAMEWORK

Fig. 2 shows the programming framework 132 in more detail. The common language specification (CLS) layer 140 supports applications written in a variety of languages 130(1), 130(2), 130(3), 130(4), ..., 130(K). Such application languages include Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python, and so on. The common language specification 140 specifies a subset of features or rules about features that, if followed, allow the various languages to communicate. For example, some languages do not support a given type (e.g., an "int\*" type) that might otherwise be supported by the common language runtime 144. In this case, the common language specification 140 does not include the type. On the other hand, types that are supported by all or most languages (e.g., the "int[]" type) is included in common language specification 140 so library developers are free to use it and are assured that the languages can handle it. This ability to communicate results in seamless integration between code modules written in one language and code modules written in another language. Since different languages are particularly well suited to particular tasks, the seamless integration between languages allows a developer to select a particular language for a particular code module with the ability to use that code module with modules written in different languages. The common language runtime 144 allow seamless multi-language development, with cross language inheritance, and provide a robust and secure execution environment for the multiple programming languages. For more information on the common language specification 140 and the common language runtime 144, the reader is directed to co-pending applications entitled "Method and System for Compiling Multiple Languages", filed 6/21/2000 (serial

number 09/598,105) and "Unified Data Type System and Method" filed 7/10/2000 (serial number 09/613,289), which are incorporated by reference.

The framework 132 encapsulates the operating system 146(1) (e.g., Windows®-brand operating systems) and object model services 146(2) (e.g., Component Object Model (COM) or Distributed COM). The operating system 146(1) provides conventional functions, such as file management, notification, event handling, user interfaces (e.g., windowing, menus, dialogs, etc.), security, authentication, verification, processes and threads, memory management, and so on. The object model services 146(2) provide interfacing with other objects to perform various tasks. Calls made to the API layer 142 are handed to the common language runtime layer 144 for local execution by the operating system 146(1) and/or object model services 146(2).

The API 142 groups API functions into multiple namespaces. Namespaces essentially define a collection of classes, interfaces, delegates, enumerations, and structures, which are collectively called "types", that provide a specific set of related functionality. A class represents managed heap allocated data that has reference assignment semantics. A delegate is an object oriented function pointer. An enumeration is a special kind of value type that represents named constants. A structure represents static allocated data that has value assignment semantics. An interface defines a contract that other types can implement.

By using namespaces, a designer can organize a set of types into a hierarchical namespace. The designer is able to create multiple groups from the set of types, with each group containing at least one type that exposes logically related functionality. In the exemplary implementation, the API 142 is organized into four root namespaces: a first namespace 200 for Web applications, a second

namespace 202 for client applications, a third namespace 204 for data and XML, and a fourth namespace 206 for base class libraries (BCLs). Each group can then be assigned a name. For instance, types in the Web applications namespace 200 are assigned the name "Web", and types in the data and XML namespace 204 can be assigned names "Data" and "XML" respectively. The named groups can be organized under a single "global root" namespace for system level APIs, such as an overall System namespace. By selecting and prefixing a top level identifier, the types in each group can be easily referenced by a hierarchical name that includes the selected top level identifier prefixed to the name of the group containing the type. For instance, types in the Web applications namespace 200 can be referenced using the hierarchical name "System.Web". In this way, the individual namespaces 200, 202, 204, and 206 become major branches off of the System namespace and can carry a designation where the individual namespaces are prefixed with a designator, such as a "System." prefix.

The Web applications namespace 200 pertains to Web based functionality, such as dynamically generated Web pages (e.g., Microsoft's Active Server Pages (ASP)). It supplies types that enable browser/server communication. The client applications namespace 202 pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional (2D), imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on.

The data and XML namespace 204 relates to connectivity to data sources and XML functionality. It supplies classes, interfaces, delegates, and enumerations that enable security, specify data types, and serialize objects into XML format documents or streams. The base class libraries (BCL) namespace

206 pertains to basic system and runtime functionality. It contains the fundamental types and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

In addition to the framework 132, programming tools 210 are provided to assist the developer in building Web services and/or applications. One example of the programming tools 200 is Visual Studio™, a multi-language suite of programming tools offered by Microsoft Corporation.

#### ROOT API NAMESPACES

Fig. 3 shows the API 142 and its four root namespaces in more detail. In one embodiment, the namespaces are identified according to a hierarchical naming convention in which strings of names are concatenated with periods. For instance, the Web applications namespace 200 is identified by the root name "System.Web". Within the "System.Web" namespace is another namespace for Web services, identified as "System.Web.Services", which further identifies another namespace for a description known as "System.Web.Services.Description". With this naming convention in mind, the following provides a general overview of selected namespaces of the API 142, although other naming conventions could be used with equal effect.

The Web applications namespace 200 ("System.Web") defines additional namespaces, including:

- A services namespace 300 ("System.Web.Services") containing classes that enable a developer to build and use Web services. The

services namespace 300 defines additional namespaces, including a description namespace 302 (“System.Web.Services.Description”) containing classes that enable a developer to publicly describe a Web service via a service description language (such as WSDL, a specification available from the W3C), a discovery namespace 304 (“System.Web.Services.Discovery”) containing classes that allow Web service consumers to locate available Web Services on a Web server, and a protocols namespace 306 (“System.Web.Services.Protocols”) containing classes that define the protocols used to transmit data across a network during communication between Web service clients and the Web service itself.

- A caching namespace 308 (“System.Web.Caching”) containing classes that enable developers to decrease Web application response time through temporarily caching frequently used resources on the server. This includes ASP.NET pages, web services, and user controls. (ASP.NET is the updated version of Microsoft’s ASP technology.) Additionally, a cache dictionary is available for developers to store frequently used resources, such as hash tables and other data structures.
- A configuration namespace 310 (“System.Web.Configuration”) containing classes that are used to read configuration data in for an application.
- A UI namespace 312 (“System.Web.UI”) containing types that allow developers to create controls and pages that will appear in Web

applications as user interfaces on a Web page. This namespace includes the control class, which provides all web based controls, whether those encapsulating HTML elements, higher level Web controls, or even custom User controls, with a common set of functionality. Also provided are classes which provide the web forms server controls data binding functionality, the ability to save the view state of a given control or page, as well as parsing functionality for both programmable and literal controls. Within the UI namespace 312 are two additional namespaces: an HTML controls namespace 314 ("System.Web.UI.HtmlControls") containing classes that permit developers to interact with types that encapsulates html 3.2 elements create HTML controls, and a Web controls namespace 316 ("System.Web.UI.WebControls") containing classes that allow developers to create higher level Web controls.

- A security namespace 318 ("System.Web.Security") containing classes used to implement security in web server applications, such as basic authentication, challenge response authentication, and role based authentication.
- A session state namespace 320 ("System.Web.SessionState") containing classes used to access session state values (i.e., data that lives across requests for the lifetime of the session) as well as session-level settings and lifetime management methods.

The client applications namespace 202 is composed of two namespaces:

- 1

2 • A windows forms namespace 322 (“System.Windows.Forms”)

3 containing classes for creating Windows®-based client applications

4 that take full advantage of the rich user interface features available in

5 the Microsoft Windows® operating system, such as the ability to

6 drag and drop screen elements. Such classes may include wrapped

7 APIs available in the Microsoft Windows® operating system that

8 are used in a windowing UI environment. Within this namespace

9 are a design namespace 324 (“System.Windows.Forms.Design”) that

10 contains classes to extend design-time support for Windows forms

11 and a component model namespace 326

12 (“System.Windows.Forms.ComponentModel”) that contains the

13 windows form implementation of the general component model

14 defined in System.ComponentModel. This namespace contains

15 designer tools, such as Visual Studio, which offer a rich experience

16 for developers at design time.
- 17 • A drawing namespace 328 (“System.Drawing”) containing classes

18 for graphics functionality. The drawing namespace 328 includes a

19 2D drawing namespace 330 (“System.Drawing.Drawing2D”) that

20 contains classes and enumerations to provide advanced 2-

21 dimensional and vector graphics functionality, an imaging

22 namespace 332 (“System.Drawing.Imaging”) that contains classes

23 for advanced imaging functionality, a printing namespace 334

24 (“System.Drawing.Printing”) that contains classes to permit

25 developers to customize printing, and a text namespace 336

(“System.Drawing.Text”) that contains classes for advanced typography functionality.

The data and XML namespace 204 is composed of two namespaces:

- A data namespace 340 (“System.Data”) containing classes that enable developers to build components that efficiently manage data from multiple data sources. It implements an architecture that, in a disconnected scenario (such as the Internet), provides tools to request, update, and reconcile data in multiple tier systems. The data namespace 340 includes a common namespace 342 that contains types shared by data providers. A data provider describes a collection of types used to access a data source, such as a database, in the managed space. The data namespace 340 also includes an OLE DB namespace 344 that contains types pertaining to data used in object-oriented databases (e.g., Microsoft’s SQL Server), and a SQL client namespace 346 that contains types pertaining to data used by SQL clients. The data namespace also includes a SQL types namespace 348 (“System.Data.SqlTypes”) that contains classes for native data types within Microsoft’s SQL Server. The classes provide a safer, faster alternative to other data types. Using the objects within this namespace helps prevent type conversion errors caused in situations where loss of precision could occur. Because other data types are converted to and from SQL types behind the



- A diagnostics namespace 364 (“System.Diagnostics”) containing classes that are used to debug applications and to trace code execution. The namespace allows developers to start system processes, read and write to event logs, and monitor system performance using performance counters.
- A globalization namespace 366 (“System.Globalization”) containing classes that define culture-related information, including the language, the country/region, the calendars in use, the format patterns for dates, currency and numbers, and the sort order for strings.
- An I/O namespace 368 (“System.IO”) containing the infrastructure pieces to operate with the input/output of data streams, files, and directories. This namespace includes a model for working with streams of bytes, higher level readers and writers which consume those bytes, various constructions or implementations of the streams (e.g., FileStream and MemoryStream) and, a set of utility classes for working with files and directories.
- A net namespace 370 (“System.Net”) providing an extensive set of classes for building network-enabled application, referred to as the Net Class Libraries (NCL). One element to the design of the Net Class Libraries is an extensible, layered approach to exposing networking functionality. The NCL stack contains three basic layers. A base layer (System.Net.Socket) provides access to an interface to TCP/IP, the communications protocol of UNIX networks and the Internet. One example of such an interface is the “WinSock

API” from Microsoft Corporation. The next layer is the Transport Protocol classes, which support such transport protocols as TCP and UDP. Developers may write their own protocol classes to provide support for protocols such as IGMP and ICMP. The third layer is the Web request, which provides an abstract factory pattern for the creation of other protocol classes. The NCL provides implementations for Hyper Text Transport Protocol (HTTP).

- A reflection namespace (“System.Reflection”) 372 containing types that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types.
- A resources namespace 374 (“System.Resources”) containing classes and interfaces that allow developers to create, store and manage various culture-specific resources used in an application.
- A security namespace 376 (“System.Security”) supporting the underlying structure of the security system, including interfaces, attributes, exceptions, and base classes for permissions.
- A service process namespace 378 (“System.ServiceProcess”) containing classes that allow developers to install and run services. Services are long-running executables that run without a user interface. They can be installed to run under a system account that enables them to be started at computer reboot. Services whose implementation is derived from processing in one class can define specific behavior for start, stop, pause, and continue commands, as well as behavior to take when the system shuts down.

- A text namespace 380 ("System.Text") containing classes representing various types of encodings (e.g., ASCII, Unicode, UTF-7, and UTF-8), abstract base classes for converting blocks of characters to and from blocks of bytes, and a helper class that manipulates and formats string objects without creating intermediate instances.
- A threading namespace 382 ("System.Threading") containing classes and interfaces that enable multi-threaded programming. The threading namespace includes a ThreadPool class that manages groups of threads, a Timer class that enables a delegate to be called after a specified amount of time, and a Mutex class for synchronizing mutually-exclusive threads. This namespace also provides classes for thread scheduling, wait notification, and deadlock resolution.
- A runtime namespace 384 ("System.Runtime") containing multiple namespaces concerning runtime features, including an interoperation services namespace 386 ("System.Runtime.InteropServices") that contains a collection of classes useful for accessing COM objects. The types in the InteropServices namespace fall into the following areas of functionality: attributes, exceptions, managed definitions of COM types, wrappers, type converters, and the Marshal class. The runtime namespace 384 further includes a remoting namespace 388 ("System.Runtime.Remoting") that contains classes and interfaces allowing developers to create and configure distributed applications. Another namespace within the runtime namespace 384 is a

serialization namespace 390 ("System.Runtime.Serialization") that contains classes used for serializing and deserializing objects. Serialization is the process of converting an object or a graph of objects into a linear sequence of bytes for either storage or transmission to another location.

The data namespace ("System.Data") contains classes that allow developers to build components to manage data from various data sources. The data namespace provides tools to request, update, and reconcile data in multiple tier systems. As discussed above, the data namespace 340 includes a common namespace 342 ("System.Data.Common"), an OLE DB namespace 344 ("System.Data.OleDb"), an SQL client namespace 346 ("System.Data.SqlClient"), and an SQL Types namespace 348 ("System.Data.SqlTypes").

The data namespace 340 contains various classes, including a constraint class that contains rules to maintain the integrity of data in a data table. A data column class provides the fundamental components for creating the schema of a data table. This schema is built by adding together one or more data column objects. A data column collection class defines the schema of a data table and determines the type of data each data column can contain. A data relation class is used to relate two data table objects to one another through data column objects.

The data namespace 340 also includes a data row class that provides a primary component of the data table. A data row collection contains the actual data for the data table. A data row change event and a data column change event occur when a change is made to a data row's value or a data column's value, respectively.

The common namespace 342 contains types shared by multiple data providers. The common namespace 342 also includes various classes, such as a data adapter class that allows for the exchange of data between a data source and a data set. A data column mapping class maps column names from a data source to corresponding column names in a data table. A data table mapping class maps data returned from a query of a data source to a data table.

The OLE DB namespace 344 includes a command builder class that automatically generates SQL statements for data table updates and a connection class that provides connections to a data source, such as a network server.

The SQL client namespace 346 also includes a command builder class. Additionally, the SQL client namespace includes a connection class that represents a unique session to an SQL server data source and a data adapter class that exchanges data between a data set and an SQL server for retrieving and saving data.

The SQL Types namespace contains classes for native data types within Microsoft's SQL Server.

Using these classes helps prevent type conversion errors caused in situations where loss of precision could occur. Other data types are converted to and from SQL types (behind the scenes), such that explicitly creating and using objects in the data namespace results in faster code. Specific details regarding the System.Data namespace are provided below.

## **System.Data**

### *Description*

The **System.Data** namespace consists mostly of the classes that constitute the ADO.NET architecture. The ADO.NET architecture enables you to build components that efficiently manage data from multiple data sources. In a disconnected scenario (such as the Internet), ADO.NET provides the tools to request, update, and reconcile data in multiple tier systems. The ADO.NET architecture is also implemented in client applications, such as Windows Forms, or HTML pages created by ASP.NET.

AcceptRejectRule enumeration (System.Data)

### *Description*

Determines the action that occurs when the **System.Data.DataSet.AcceptChanges** or **System.Data.DataTable.RejectChanges** method is invoked on a **System.Data.DataTable** with a **System.Data.ForeignKeyConstraint**.

Changes to a **System.Data.DataTable** are not final until you call the **System.Data.DataTable.AcceptChanges** method. At that time, constraint-related errors can occur because any **System.Data.ForeignKeyConstraint** objects associated with a **System.Data.DataTable** are activated to allow deletions and edits to occur freely. Prior to that time, **System.Data.ForeignKeyConstraint** objects are inactive. When the **System.Data.ForeignKeyConstraint** becomes activated, and errors occur, **System.Data.AcceptRejectRule** is called to determine the next course of action.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const AcceptRejectRule Cascade;
[C++] public: const AcceptRejectRule Cascade;
[VB] Public Const Cascade As AcceptRejectRule
[JScript] public var Cascade : AcceptRejectRule;
```

*Description*

Changes are cascaded across the relationship.

```
[C#] public const AcceptRejectRule None;
[C++] public: const AcceptRejectRule None;
[VB] Public Const None As AcceptRejectRule
[JScript] public var None : AcceptRejectRule;
```

*Description*

No action occurs.

Methods:

CommandBehavior enumeration (System.Data)

ToString

*Description*

Specifies a description of the results and the affect on the database of the query command.

The **System.Data.CommandBehavior** values are used by the **System.Data.IDbCommand.ExecuteReader** method of **System.Data.IDbCommand** and any classes derived from it.

**ToString**

[C#] public const CommandBehavior CloseConnection;  
[C++] public: const CommandBehavior CloseConnection;  
[VB] Public Const CloseConnection As CommandBehavior  
[JScript] public var CloseConnection : CommandBehavior;

#### *Description*

When the command is executed, the associated **Connection** object is closed when the associated **DataReader** object is closed.

**ToString**

[C#] public const CommandBehavior Default;  
[C++] public: const CommandBehavior Default;  
[VB] Public Const Default As CommandBehavior  
[JScript] public var Default : CommandBehavior;

**ToString**

[C#] public const CommandBehavior KeyInfo;  
[C++] public: const CommandBehavior KeyInfo;  
[VB] Public Const KeyInfo As CommandBehavior  
[JScript] public var KeyInfo : CommandBehavior;

## *Description*

The query returns column and primary key information. The query is executed without any locking on the selected rows. When using **System.Data.CommandBehavior.KeyInfo**, the SQL Server .NET Data Provider appends a FOR BROWSE clause to the statement being executed. The user should be aware of potential side effects, such as interference with the use of SET FMTONLY ON statements. See SQL Server Books Online for more information.

### *ToString*

```
[C#] public const CommandBehavior SchemaOnly;
[C++] public: const CommandBehavior SchemaOnly;
[VB] Public Const SchemaOnly As CommandBehavior
[JScript] public var SchemaOnly : CommandBehavior;
```

## *Description*

The query returns column information only and does not affect the database state.

### *ToString*

```
[C#] public const CommandBehavior SequentialAccess;
[C++] public: const CommandBehavior SequentialAccess;
[VB] Public Const SequentialAccess As CommandBehavior
[JScript] public var SequentialAccess : CommandBehavior;
```

*Description*

The results of the query are read sequentially to the column level. This allows an application to read large binary values using the **GetChars** or **GetBytes** methods of a .NET data provider. Execution of the query may affect the database state.

**ToString**

[C#] public const CommandBehavior SingleResult;  
 [C++] public: const CommandBehavior SingleResult;  
 [VB] Public Const SingleResult As CommandBehavior  
 [JScript] public var SingleResult : CommandBehavior;

*Description*

The query returns a single result. Execution of the query may affect the database state.

**ToString**

[C#] public const CommandBehavior SingleRow;  
 [C++] public: const CommandBehavior SingleRow;  
 [VB] Public Const SingleRow As CommandBehavior  
 [JScript] public var SingleRow : CommandBehavior;

*Description*

The query is expected to return a single row. Execution of the query may affect the database state. Some .NET data providers may, but are not required to, use this information to optimize the performance of the command. When you specify **System.Data.CommandBehavior.SingleRow** with the **System.Data.OleDb.OleDbCommand.ExecuteReader** method of the **System.Data.OleDb.OleDbCommand** object, the OLE DB .NET Data Provider performs binding using the OLE DB IRow interface if it is available. Otherwise, it uses the IRowset interface. If your SQL statement is expected to return only a single row, specifying **System.Data.CommandBehavior.SingleRow** can also improve application performance.

CommandType enumeration (System.Data)

ToString

### *Description*

Specifies how a command string is interpreted.

When the **System.Data.IDbCommand.CommandType** property is set to **StoredProcedure**, set the **System.Data.IDbCommand.CommandText** property to the name of the stored procedure. The command executes this stored procedure when you call **System.Data.IDbCommand.ExecuteReader**.

ToString

[C#] public const CommandType StoredProcedure;

[C++] public: const CommandType StoredProcedure;

[VB] Public Const StoredProcedure As CommandType

1 [JScript] public var StoredProcedure : CommandType;

3 *Description*

4 The name of a stored procedure.

5 ToString

7 [C#] public const CommandType TableDirect;

8 [C++] public: const CommandType TableDirect;

9 [VB] Public Const TableDirect As CommandType

10 [JScript] public var TableDirect : CommandType;

12 *Description*

13 A table name whose columns are all returned (OLE DB .NET Data  
14 Provider only).

15 ToString

17 [C#] public const CommandType Text;

18 [C++] public: const CommandType Text;

19 [VB] Public Const Text As CommandType

20 [JScript] public var Text : CommandType;

22 *Description*

23 A SQL text command.

24 ConnectionState enumeration (System.Data)

25 ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Returns the current state of the connection to a data source.

The **System.Data.ConnectionState** values are used by the **System.Data.OleDb.OleDbConnection.State** property of the **System.Data.OleDb.OleDbConnection** and **System.Data.SqlClient.SqlConnection** objects.

*ToString*

[C#] public const ConnectionState Broken;

[C++] public: const ConnectionState Broken;

[VB] Public Const Broken As ConnectionState

[JScript] public var Broken : ConnectionState;

*Description*

The object is broken. This can occur only after the connection has been opened. A connection in this state may be closed and then re-opened.

*ToString*

[C#] public const ConnectionState Closed;

[C++] public: const ConnectionState Closed;

[VB] Public Const Closed As ConnectionState

[JScript] public var Closed : ConnectionState;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

The object is closed.  
ToString

```
[C#] public const ConnectionState Connecting;
[C++] public: const ConnectionState Connecting;
[VB] Public Const Connecting As ConnectionState
[JScript] public var Connecting : ConnectionState;
```

*Description*

The object is connecting.  
ToString

```
[C#] public const ConnectionState Executing;
[C++] public: const ConnectionState Executing;
[VB] Public Const Executing As ConnectionState
[JScript] public var Executing : ConnectionState;
```

*Description*

The object is executing a command.  
ToString

```
[C#] public const ConnectionState Fetching;
[C++] public: const ConnectionState Fetching;
```

1 [VB] Public Const Fetching As ConnectionState

2 [JScript] public var Fetching : ConnectionState;

3

4 *Description*

5 Data is being retrieved.

6 ToString

7

8 [C#] public const ConnectionState Open;

9 [C++] public: const ConnectionState Open;

10 [VB] Public Const Open As ConnectionState

11 [JScript] public var Open : ConnectionState;

12

13 *Description*

14 The object is open.

15 Constraint class (System.Data)

16 ToString

17

18

19 *Description*

20 Represents a constraint that can be enforced on one or more

21 **System.Data.DataColumn** objects.

22 A constraint is a rule used to maintain the integrity of the data in the

23 **System.Data.DataTable** . For example, when you delete a value that is used in

24 one or more related tables, a **System.Data.ForeignKeyConstraint** determines

25 whether the values in the related tables are also deleted, set to null values, set to

default values, or whether no action occurs. A **System.Data.UniqueConstraint** ,  
on the other hand, simply ensures that all values within a particular table are  
unique. For more information, see .

Constructors:

Constraint

*Example Syntax:*

ToString

[C#] protected Constraint();

[C++] protected: Constraint();

[VB] Protected Sub New()

[JScript] protected function Constraint();

Properties:

\_DataSet

ToString

[C#] protected internal virtual DataSet \_DataSet {get;}

[C++] internal: \_\_property virtual DataSet\* get \_DataSet();

[VB] Overridable Protected Friend ReadOnly Property \_DataSet As DataSet

[JScript] package function get \_DataSet() : DataSet;

### *Description*

Gets the **System.Data.DataSet** to which this constraint belongs.

ConstraintName

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public virtual string ConstraintName {get; set;}
[C++] public: __property virtual String* get_ConstraintName();public: __property
virtual void set_ConstraintName(String*);
[VB] Overridable Public Property ConstraintName As String
[JScript] public function get ConstraintName() : String;public function set
ConstraintName(String);
```

*Description*

The name of a constraint in the **System.Data.ConstraintCollection** .

The **System.Data.ConstraintCollection** is returned by the **System.Data.DataTable.Constraints** property of the **System.Data.DataTable** class.

ExtendedProperties

ToString

```
[C#] public PropertyCollection ExtendedProperties {get;}
[C++] public: __property PropertyCollection* get_ExtendedProperties();
[VB] Public ReadOnly Property ExtendedProperties As PropertyCollection
[JScript] public function get ExtendedProperties() : PropertyCollection;
```

*Description*

Gets the collection of customized user information.

1 Use the **System.Data.DataTable.ExtendedProperties** to add custom  
2 information to a **System.Data.DataTable** . Add information with the Add method.  
3 Retrieve information with the Item method.

4 Table

5 ToString

6  
7 [C#] public abstract DataTable Table {get;}

8 [C++] public: \_\_property virtual DataTable\* get\_Table() = 0;

9 [VB] MustOverride Public ReadOnly Property Table As DataTable

10 [JScript] public abstract function get Table() : DataTable;

11  
12 *Description*

13 Gets the **System.Data.DataTable** to which the constraint applies.

14 CheckStateForProperty

15  
16 [C#] protected void CheckStateForProperty();

17 [C++] protected: void CheckStateForProperty();

18 [VB] Protected Sub CheckStateForProperty()

19 [JScript] protected function CheckStateForProperty();

20  
21 *Description*

22 SetDataSet

23  
24 [C#] protected internal void SetDataSet(DataSet dataSet);

25 [C++] protected public: void SetDataSet(DataSet\* dataSet);

1 [VB] Protected Friend Dim Sub SetDataSet(ByVal dataSet As DataSet)

2 [JScript] package function SetDataSet(dataSet : DataSet);

3

4 *Description*

5       Sets the constraint's **System.Data.DataSet** . The **System.Data.DataSet** to  
6 which this constraint will belong.

7       ToString

8

9 [C#] public override string ToString();

10 [C++] public: String\* ToString();

11 [VB] Overrides Public Function ToString() As String

12 [JScript] public override function ToString() : String;

13

14 *Description*

15       Gets the **System.Data.Constraint.ConstraintName** , if there is one, as a  
16 string.

17 *Return Value:* The string value of the **System.Data.Constraint.ConstraintName**

18

19       ConstraintCollection class (System.Data)

20       ToString

21

22

23 *Description*

24       Represents a collection of constraints for a **System.Data.DataTable** .

25

100% Satisfied

The **System.Data.ConstraintCollection** is accessed through the **System.Data.DataTable.Constraints** property.

Count

IsReadOnly

IsSynchronized

Item

ToString

#### *Description*

Gets the **System.Data.Constraint** from the collection with the specified name.

The **System.Data.ConstraintCollection.Contains(System.String)** method can be used to test for the existence of a specific constraint. The **System.Data.Constraint.ConstraintName** of the constraint to return.

Item

ToString

[C#] public virtual Constraint this[int index] {get;}

[C++] public: \_\_property virtual Constraint\* get\_Item(int index);

[VB] Overridable Public Default ReadOnly Property Item(ByVal index As

Integer) As Constraint

[JScript] returnValue = ConstraintCollectionObject.Item(index); Gets the specified **System.Data.Constraint**.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Gets the **System.Data.Constraint** from the collection at the specified index.

The **System.Data.ConstraintCollection.Contains(System.String)** method can be used to test for the existence of a specific constraint. The index of the constraint to return.

List

ToString

[C#] protected override ArrayList List {get;}

[C++] protected: \_\_property virtual ArrayList\* get\_List();

[VB] Overrides Protected ReadOnly Property List As ArrayList

[JScript] protected function get List() : ArrayList;

*Description*

Gets the list of objects contained by the collection.

SyncRoot

ToString

*Description*

Occurs when the **System.Data.ConstraintCollection** is changed through additions or removals.

For more information about handling events, see .

## Add

[C#] public void Add(Constraint constraint);

[C++] public: void Add(Constraint\* constraint);

[VB] Public Sub Add(ByVal constraint As Constraint)

[JScript] public function Add(constraint : Constraint); Adds a constraint to the

**System.Data.ConstraintCollection** .

### *Description*

Adds the specified constraint to the collection.

If the collection is successfully changed by adding or removing constraints, the **System.Data.ConstraintCollection.CollectionChanged** event occurs. The **System.Data.Constraint** to add.

## Add

[C#] public virtual Constraint Add(string name, DataColumn column, bool primaryKey);

[C++] public: virtual Constraint\* Add(String\* name, DataColumn\* column, bool primaryKey);

[VB] Overridable Public Function Add(ByVal name As String, ByVal column As DataColumn, ByVal primaryKey As Boolean) As Constraint

[JScript] public function Add(name : String, column : DataColumn, primaryKey : Boolean) : Constraint;

### *Description*

Constructs a new **System.Data.UniqueConstraint** , using the specified **System.Data.DataColumn** , and adds it to the collection.

The **System.Data.ConstraintCollection.CollectionChanged** event occurs if the constraint is added successfully. The name of the **System.Data.UniqueConstraint**. The **System.Data.DataColumn** affected by the constraint. Indicates whether the column is a primary key column.

#### Add

[C#] public virtual Constraint Add(string name, DataColumn primaryKeyColumn, DataColumn foreignKeyColumn);

[C++] public: virtual Constraint\* Add(String\* name, DataColumn\* primaryKeyColumn, DataColumn\* foreignKeyColumn);

[VB] Overridable Public Function Add(ByVal name As String, ByVal primaryKeyColumn As DataColumn, ByVal foreignKeyColumn As DataColumn) As Constraint

[JScript] public function Add(name : String, primaryKeyColumn : DataColumn, foreignKeyColumn : DataColumn) : Constraint;

#### *Description*

Constructs a new **System.Data.ForeignKeyConstraint** , with the specified parent and child columns, and adds the constraint to the collection.

A **System.Data.ForeignKeyConstraint** and **System.Data.UniqueConstraint** are both created and added automatically when a **System.Data.DataRelation** is added to a **System.Data.DataSet** object's **System.Data.DataRelationCollection** . The

**System.Data.ForeignKeyConstraint** (which gets the same name as the  
**System.Data.DataRelation** ) is added to the child table's  
**System.Data.ConstraintCollection** , and the **System.Data.UniqueConstraint** is  
 added to the parent table's **System.Data.ConstraintCollection** . The name of the  
**System.Data.UniqueConstraint**. The primary key **System.Data.DataColumn**.  
 The foreign key **System.Data.DataColumn**.

Add

```

[C#] public virtual Constraint Add(string name, DataColumn[] columns, bool
primaryKey);
[C++] public: virtual Constraint* Add(String* name, DataColumn* columns[],
bool primaryKey);
[VB] Overridable Public Function Add(ByVal name As String, ByVal columns()
As DataColumn, ByVal primaryKey As Boolean) As Constraint
[JScript] public function Add(name : String, columns : DataColumn[],
primaryKey : Boolean) : Constraint;
    
```

### *Description*

Constructs a new **System.Data.UniqueConstraint** using the specified  
 array of **System.Data.DataColumn** objects, and adds it to the collection.

The **System.Data.ConstraintCollection.CollectionChanged** event occurs  
 if the constraint is added successfully. The name of the  
**System.Data.UniqueConstraint**. An array of **System.Data.DataColumn** objects  
 that are affected by the constraint. Indicates whether the columns are primary key  
 columns.

## Add

```
[C#] public virtual Constraint Add(string name, DataColumn[]
primaryKeyColumns, DataColumn[] foreignKeyColumns);
[C++] public: virtual Constraint* Add(String* name, DataColumn*
primaryKeyColumns[], DataColumn* foreignKeyColumns[]);
[VB] Overridable Public Function Add(ByVal name As String, ByVal
primaryKeyColumns() As DataColumn, ByVal foreignKeyColumns() As
DataColumn) As Constraint
[JScript] public function Add(name : String, primaryKeyColumns : DataColumn[],
foreignKeyColumns : DataColumn[]) : Constraint;
```

### *Description*

Constructs a new **System.Data.ForeignKeyConstraint** , with the specified parent columns and child columns, and adds the constraint to the collection.

A **System.Data.ForeignKeyConstraint** and a **System.Data.UniqueConstraint** are created automatically when you add a **System.Data.DataRelation** to a **System.Data.DataSet** . In that case, adding a second **System.Data.ForeignKeyConstraint** on the same columns will result in an exception. To avoid this, use the **System.Data.ForeignKeyConstraint** constructor to create the **System.Data.ForeignKeyConstraint** and test it against existing collection members with the **System.Data.ForeignKeyConstraint.Equals(System.Object)** method. The name of the **System.Data.UniqueConstraint**. An array of **System.Data.DataColumn**

objects that are the primary key columns. An array of **System.Data.DataColumn** objects that are the foreign key columns.

### AddRange

```
[C#] public void AddRange(Constraint[] constraints);  
[C++] public: void AddRange(Constraint* constraints[]);  
[VB] Public Sub AddRange(ByVal constraints() As Constraint)  
[JScript] public function AddRange(constraints : Constraint[]);
```

### *Description*

Copies the elements of the specified **System.Data.ConstraintCollection** array to the end of the collection. An array of **System.Data.ConstraintCollection** objects to add to the collection.

### CanRemove

```
[C#] public bool CanRemove(Constraint constraint);  
[C++] public: bool CanRemove(Constraint* constraint);  
[VB] Public Function CanRemove(ByVal constraint As Constraint) As Boolean  
[JScript] public function CanRemove(constraint : Constraint) : Boolean;
```

### *Description*

Indicates if a **System.Data.Constraint** can be removed.

*Return Value:* Generates an exception if the **System.Data.Constraint** can't be removed from collection. Otherwise, **true** if the **System.Data.Constraint** can be removed.

When a **System.Data.DataRelation** is added to a **System.Data.DataSet** , a **System.Data.ForeignKeyConstraint** and **System.Data.UniqueConstraint** are added automatically to the parent table and the child table. The **System.Data.UniqueConstraint** is applied to the parent table's primary key column, and the **System.Data.ForeignKeyConstraint** is applied to the child table's foreign key column. In that case, attempting to remove the **System.Data.UniqueConstraint** will cause an exception to be thrown because the **System.Data.ForeignKeyConstraint** must be removed first. To avoid this, use the **System.Data.ConstraintCollection.CanRemove(System.Data.Constraint)** to determine if a **System.Data.UniqueConstraint** can be removed. The **System.Data.Constraint** to be tested for removal from the collection.

Clear

[C#] public void Clear();

[C++] public: void Clear();

[VB] Public Sub Clear()

[JScript] public function Clear();

### *Description*

Clears the collection of any **System.Data.Constraint** objects.

The **System.Data.ConstraintCollection.CollectionChanged** occurs if this action is successful.

Contains

[C#] public bool Contains(string name);

1 [C++] public: bool Contains(String\* name);

2 [VB] Public Function Contains(ByVal name As String) As Boolean

3 [JScript] public function Contains(name : String) : Boolean;

4  
5 *Description*

6 Indicates whether the **System.Data.Constraint** , specified by name, exists  
7 in the collection.

8 *Return Value:* **true** if the collection contains the specified constraint; otherwise,  
9 **false** .

10 Use the **System.Data.ConstraintCollection.Contains(System.String)**  
11 method to determine if the specified **System.Data.Constraint** exists before  
12 attempting to remove it from the collection. You can also use the  
13 **System.Data.ConstraintCollection.CanRemove(System.Data.Constraint)**  
14 method to determine if a **System.Data.Constraint** can be removed. The  
15 **System.Data.Constraint.ConstraintName** of the constraint.

16 *IndexOf*

17  
18 [C#] public int IndexOf(Constraint constraint);

19 [C++] public: int IndexOf(Constraint\* constraint);

20 [VB] Public Function IndexOf(ByVal constraint As Constraint) As Integer

21 [JScript] public function IndexOf(constraint : Constraint) : int;

22  
23 *Description*

24

25

Gets the index of the specified **System.Data.Constraint** .

*Return Value:* The index of the **System.Data.Constraint** if it is in the collection; otherwise, -1.

Use the

**System.Data.ConstraintCollection.IndexOf(System.Data.Constraint)** method

to return an index to be used with either the

**System.Data.ConstraintCollection.Contains(System.String)** or

**System.Data.ConstraintCollection.Remove(System.Data.Constraint)** method.

The **System.Data.Constraint** to search for.

**IndexOf**

[C#] public virtual int IndexOf(string constraintName);

[C++] public: virtual int IndexOf(String\* constraintName);

[VB] Overridable Public Function IndexOf(ByVal constraintName As String) As

Integer

[JScript] public function IndexOf(constraintName : String) : int; Gets the index of the specified **System.Data.Constraint** .

### *Description*

Gets the index of the **System.Data.Constraint** , specified by name.

*Return Value:* The index of the **System.Data.Constraint** if it is in the collection; otherwise, -1.

Use the

**System.Data.ConstraintCollection.IndexOf(System.Data.Constraint)** method

to return an index to be used with either the

1 **System.Data.ConstraintCollection.Contains(System.String)** or

2 **System.Data.ConstraintCollection.Remove(System.Data.Constraint)** method.

3 The name of the **System.Data.Constraint** .

4       **OnCollectionChanged**

5  
6 [C#] protected virtual void OnCollectionChanged(CollectionChangeEventArgs  
7 ccevent);

8 [C++] protected: virtual void OnCollectionChanged(CollectionChangeEventArgs\*  
9 ccevent);

10 [VB] Overridable Protected Sub OnCollectionChanged(ByVal ccevent As  
11 CollectionChangeEventArgs)

12 [JScript] protected function OnCollectionChanged(ccevent :  
13 CollectionChangeEventArgs);

14  
15 *Description*

16       Raises the **System.Data.ConstraintCollection.CollectionChanged** event.

17       Raising an event invokes the event handler through a delegate. For more  
18 information, see . A **System.ComponentModel.CollectionChangeEventArgs**  
19 that contains the event data.

20       **Remove**

21  
22 [C#] public void Remove(Constraint constraint);

23 [C++] public: void Remove(Constraint\* constraint);

24 [VB] Public Sub Remove(ByVal constraint As Constraint)

25 [JScript] public function Remove(constraint : Constraint); Removes a

## 1 **System.Data.Constraint** from the **System.Data.ConstraintCollection** .

### 3 *Description*

4 Removes the specified **System.Data.Constraint** from the collection.

5 Use the **System.Data.ConstraintCollection.Contains(System.String)**  
6 method to determine if the collection contains the target **System.Data.Constraint**  
7 . The **System.Data.Constraint** to remove.

8 Remove

10 [C#] public void Remove(string name);

11 [C++] public: void Remove(String\* name);

12 [VB] Public Sub Remove(ByVal name As String)

13 [JScript] public function Remove(name : String);

### 15 *Description*

16 Removes the constraint, specified by name, from the collection.

17 Use the **System.Data.ConstraintCollection.Contains(System.String)**  
18 method to determine if the collection contains the target **System.Data.Constraint**  
19 . The name of the **System.Data.Constraint** to remove.

20 RemoveAt

22 [C#] public void RemoveAt(int index);

23 [C++] public: void RemoveAt(int index);

24 [VB] Public Sub RemoveAt(ByVal index As Integer)

25 [JScript] public function RemoveAt(index : int);

1  
2 *Description*

3 Removes the constraint at the specified index from the collection.

4 The

5 **System.Data.ConstraintCollection.IndexOf(System.Data.Constraint)** method

6 returns the index of a given **System.Data.Constraint** . The index of the

7 **System.Data.Constraint** to remove.

8 ConstraintException class (System.Data)

9 ToString

10  
11  
12 *Description*

13 Represents the exception that is thrown when attempting an action that  
14 violates a constraint.

15 ConstraintException

16 *Example Syntax:*

17 ToString

18  
19 [C#] public ConstraintException();

20 [C++] public: ConstraintException();

21 [VB] Public Sub New()

22 [JScript] public function ConstraintException(); Initializes a new instance of the

23 **System.Data.ConstraintException** class.

24  
25 *Description*

Initializes a new instance of the **System.Data.ConstraintException** class.

**ConstraintException**

*Example Syntax:*

**ToString**

[C#] public ConstraintException(string s);

[C++] public: ConstraintException(String\* s);

[VB] Public Sub New(ByVal s As String)

[JScript] public function ConstraintException(s : String);

*Description*

Initializes a new instance of the **System.Data.ConstraintException** class with the specified string. The string to display when the exception is thrown.

**ConstraintException**

*Example Syntax:*

**ToString**

[C#] public ConstraintException(SerializationInfo info, StreamingContext context);

[C++] public: ConstraintException(SerializationInfo\* info, StreamingContext context);

[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] public function ConstraintException(info : SerializationInfo, context : StreamingContext); Initializes a new instance of the

**System.Data.ConstraintException** class.

*Description*

Initializes a new instance of the **System.Data.ConstraintException** class.  
The data necessary to serialize or deserialize an object. Description of the source and destination of the specified serialized stream.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

DataColumn class (System.Data)

ToString

*Description*

Represents the schema of a column in a **System.Data.DataTable** .

The **System.Data.DataColumn** is the fundamental building block for creating the schema of a **System.Data.DataTable** . You build the schema by adding one or more **System.Data.DataColumn** objects to the **System.Data.DataColumnCollection** . For more information, see .

DataColumn

*Example Syntax:*

## ToString

[C#] public DataColumn();

[C++] public: DataColumn();

[VB] Public Sub New()

[JScript] public function DataColumn(); Initializes a new instance of the

**System.Data.DataColumn** class.

### *Description*

Initializes a new instance of a **System.Data.DataColumn** class.

When created, a new **System.Data.DataColumn** object has no default

**System.Data.DataColumn.ColumnName** or

**System.Data.DataColumn.Caption** . When added to a

**System.Data.DataColumnCollection** , however, a default name ("Column1",

"Column2", etc.) is given to the column.

**DataColumn**

### *Example Syntax:*

## ToString

[C#] public DataColumn(string columnName);

[C++] public: DataColumn(String\* columnName);

[VB] Public Sub New(ByVal columnName As String)

[JScript] public function DataColumn(columnName : String);

### *Description*

1        Initializes a new instance of the **System.Data.DataColumn** class using the  
2 specified column name.

3        By default, the name given to a column becomes the  
4 **System.Data.DataColumn.Caption** property value. A string that represents the  
5 name of the column to be created. If set to **null** or an empty string (""), a default  
6 name will be given when added to the columns collection.

7        DataColumn

8        *Example Syntax:*

9        ToString

11      [C#] public DataColumn(string columnName, Type dataType);

12      [C++] public: DataColumn(String\* columnName, Type\* dataType);

13      [VB] Public Sub New(ByVal columnName As String, ByVal dataType As Type)

14      [JScript] public function DataColumn(columnName : String, dataType : Type);

16      *Description*

17        Initializes a new instance of the **System.Data.DataColumn** class using the  
18 specified column name and data type. A string that represents the name of the  
19 column to be created. If set to **null** or an empty string (""), a default name will be  
20 given when added to the columns collection. A supported

21 **System.Data.DataColumn.DataType** .

22        DataColumn

23        *Example Syntax:*

24        ToString

```

1
2 [C#] public DataColumn(string columnName, Type dataType, string expr);
3 [C++] public: DataColumn(String* columnName, Type* dataType, String* expr);
4 [VB] Public Sub New(ByVal columnName As String, ByVal dataType As Type,
5   ByVal expr As String)
6 [JScript] public function DataColumn(columnName : String, dataType : Type,
7   expr : String);
8

```

### 9 *Description*

10        Initializes a new instance of the **System.Data.DataColumn** class using the  
11 specified name, data type, and expression. A string that represents the name of the  
12 column to be created. If set to **null** or an empty string (""), a default name will be  
13 given when added to the columns collection. A supported  
14 **System.Data.DataColumn.DataType** . The expression used to create this  
15 column. For more details, see the **System.Data.DataColumn.Expression**  
16 property.

17        DataColumn

18        *Example Syntax:*

19        ToString

```

20
21 [C#] public DataColumn(string columnName, Type dataType, string expr,
22   MappingType type);
23 [C++] public: DataColumn(String* columnName, Type* dataType, String* expr,
24   MappingType type);
25 [VB] Public Sub New(ByVal columnName As String, ByVal dataType As Type,

```

```

1 ByVal expr As String, ByVal type As MappingType)
2 [JScript] public function DataColumn(columnName : String, dataType : Type,
3 expr : String, type : MappingType);
4

```

#### 5 *Description*

6        Initializes a new instance of the **System.Data.DataColumn** class using the  
7 specified name, data type, expression, and value that determines whether the  
8 column is an attribute.

9        The *type* argument sets the **System.Data.DataColumn.ColumnMapping**  
10 property. The property specifies how a **System.Data.DataColumn** is mapped  
11 when a **System.Data.DataSet** is transformed into an XML document. For  
12 example, if the the column is named "fName," and the value it contains is "Bob,"  
13 and *type* is set to **MappingType.Attribute** , the XML element would be: For  
14 more details on how columns are mapped to elements or attributes, see the  
15 **System.Data.DataColumn.ColumnMapping** property. A string that represents  
16 the name of the column to be created. If set to **null** or an empty string (""), a  
17 default name will be given when added to the columns collection. A supported  
18 **System.Data.DataColumn.DataType** . The expression used to create this  
19 column. For more details, see the **System.Data.DataColumn.Expression**  
20 property. One of the **System.Data.MappingType** values.

21        AllowDBNull

22        ToString

24 [C#] public bool AllowDBNull {get; set;}

25 [C++] public: \_\_property bool get\_AllowDBNull();public: \_\_property void

1 set\_AllowDBNull(bool);

2 [VB] Public Property AllowDBNull As Boolean

3 [JScript] public function get AllowDBNull() : Boolean;public function set

4 AllowDBNull(Boolean);

6 *Description*

7 Gets or sets a value indicating whether null values are allowed in this  
8 column for rows belonging to the table.

9 AutoIncrement

10 ToString

12 [C#] public bool AutoIncrement {get; set;}

13 [C++] public: \_\_property bool get \_AutoIncrement();public: \_\_property void

14 set \_AutoIncrement(bool);

15 [VB] Public Property AutoIncrement As Boolean

16 [JScript] public function get AutoIncrement() : Boolean;public function set

17 AutoIncrement(Boolean);

19 *Description*

20 Gets or sets a value indicating whether the column automatically  
21 increments the value of the column for new rows added to the table.

22 If the type of this column is not Int16, Int32, or Int64 when this property is  
23 set, the **System.Data.DataColumn.DataType** property is coerced to Int32. An  
24 exception is generated if this is a computed column (that is, the  
25 **System.Data.DataColumn.Expression** property is set.) The incremented value is

used only if the row's value for this column, when added to the columns collection, is equal to the default value.

AutoIncrementSeed

ToString

```
[C#] public long AutoIncrementSeed {get; set;}
```

```
[C++] public: __property __int64 get_AutoIncrementSeed();public: __property  
void set_AutoIncrementSeed(__int64);
```

```
[VB] Public Property AutoIncrementSeed As Long
```

```
[JScript] public function get AutoIncrementSeed() : long;public function set  
AutoIncrementSeed(long);
```

### *Description*

Gets or sets the starting value for a column that has its

**System.Data.DataColumn.AutoIncrement** property set to **true** .

AutoIncrementStep

ToString

```
[C#] public long AutoIncrementStep {get; set;}
```

```
[C++] public: __property __int64 get_AutoIncrementStep();public: __property  
void set_AutoIncrementStep(__int64);
```

```
[VB] Public Property AutoIncrementStep As Long
```

```
[JScript] public function get AutoIncrementStep() : long;public function set  
AutoIncrementStep(long);
```

1  
2 *Description*

3 Gets or sets the increment used by a column with its

4 **System.Data.DataColumn.AutoIncrement** property set to **true** .

5 Caption

6 ToString

7  
8 [C#] public string Caption {get; set;}

9 [C++] public: \_\_property String\* get\_Caption();public: \_\_property void  
10 set\_Caption(String\*);

11 [VB] Public Property Caption As String

12 [JScript] public function get Caption() : String;public function set Caption(String);

13  
14 *Description*

15 Gets or sets the caption for the column.

16 The **System.Data.DataColumn.Caption** value becomes visible in controls  
17 that support its display. For example, the **System.Windows.Forms.DataGrid**  
18 control is capable of displaying captions for each column.

19 ColumnMapping

20 ToString

21  
22 [C#] public virtual MappingType ColumnMapping {get; set;}

23 [C++] public: \_\_property virtual MappingType get\_ColumnMapping();public:  
24 \_\_property virtual void set\_ColumnMapping(MappingType);

25 [VB] Overridable Public Property ColumnMapping As MappingType

[JScript] public function get ColumnMapping() : MappingType;public function set ColumnMapping(MappingType);

#### *Description*

Gets or sets the **System.Data.MappingType** of the column.

The **System.Data.DataColumn.ColumnMapping** property determines how a **System.Data.DataColumn** is mapped when a **System.Data.DataSet** is saved as an XML document using the

**System.Data.DataSet.WriteXml(System.IO.Stream)** method.

ColumnName

ToString

[C#] public string ColumnName {get; set;}

[C++] public: \_\_property String\* get\_ColumnName();public: \_\_property void set\_ColumnName(String\*);

[VB] Public Property ColumnName As String

[JScript] public function get ColumnName() : String;public function set ColumnName(String);

#### *Description*

Gets or sets the name of the column in the

**System.Data.DataColumnCollection** .

When a **System.Data.DataColumn** is created, it has no

**System.Data.DataColumn.ColumnName** value. When the

**System.Data.DataColumn** is added to a **System.Data.DataTable** object's

**System.Data.DataColumnCollection** , however, it is given a default name ("Column1", "Column2", etc.).

Container

DataType

ToString

### *Description*

Gets or sets the type of data stored in the column.

Setting the **System.Data.DataColumn.DataType** value is critical to ensuring the correct creation and updating of data in a DBMS.

DefaultValue

ToString

[C#] public object DefaultValue {get; set;}

[C++] public: \_\_property Object\* get\_DefaultValue();public: \_\_property void set\_DefaultValue(Object\*);

[VB] Public Property DefaultValue As Object

[JScript] public function get DefaultValue() : Object;public function set DefaultValue(Object);

### *Description*

Gets or sets the default value for the column when creating new rows.

A default value is the value that is automatically assigned to the column when a **System.Data.DataRow** is created. By setting a default value, you can give

the user an idea of what information to input. On the other hand, you can use the **System.Data.DataColumn.DefaultValue** property to automatically insert a value that shouldn't be touched by the user; for example, the current date and time of the row's creation.

DesignMode

Events

Expression

ToString

### *Description*

Gets or sets the expression used to filter rows, calculate the values in a column, or create an aggregate column.

One use of the **System.Data.DataColumn.Expression** property is to create calculated columns. For example, to calculate a tax value, the unit price is multiplied by a tax rate of a given region. Since tax rates vary from region to region, it would be impossible to put a single tax rate in a column; instead, the value is calculated using the **System.Data.DataColumn.Expression** property, as shown in the Visual Basic code below:

```
DataSet1.Tables("Products").Columns("tax").Expression = "UnitPrice * 0.086" A
```

second use is to create an aggregate column. Similar to a calculated value, an aggregate performs an operation based on the entire set of rows in the

**System.Data.DataTable**. A simple example is to count the number of rows returned in the set, which is the method you would use to count the number of transactions completed by a particular salesperson, as shown in this Visual Basic

code: DataSet1.Tables("Orders").Columns("OrderCount").Expression =  
 "Count(OrderID)" **EXPRESSION SYNTAX** When creating an expression, use the  
**System.Data.DataColumn.ColumnName** property to refer to columns. For  
 example, if the **System.Data.DataColumn.ColumnName** for one column is  
 "UnitPrice", and another "Quantity", the expression would be: "UnitPrice \*  
 Quantity" When creating an expression for a filter, enclose strings with single  
 quotes: "LastName = 'Jones'" The following characters are special characters and  
 must be escaped, as explained below, if they are to be used in a column name: \n  
 (newline) \t (tab) \r (carriage return) ~ ( ) # \ / = > < + - \* % & | ^ ' " [ ] If a column  
 name contains one of the above characters, the name must be wrapped in brackets.  
 For example to use a column named "Column#" in an expression, you would write  
 "[Column#]": Total \* [Column#] Because brackets are special characters, you  
 must use a slash ("\") to escape the bracket, if it is part of a column name. For  
 example, a column named "Column[]" would be written: Total \* [Column[\]]  
 (Only the second bracket must be escaped.) **USER-DEFINED VALUES** User-  
 defined values may be used within expressions to be compared against column  
 values. String values should be enclosed within single quotes. Date values should  
 be enclosed within pound signs (#). Decimals and scientific notation are  
 permissible for numeric values. For example: "FirstName = 'John'" "Price <=  
 50.00" "Birthdate < #1/31/82#" For columns that contain enumeration values, cast  
 the value to an integer data type. For example: "EnumColumn = 5" **OPERATORS**  
 Concatenation is allowed using Boolean AND, OR, and NOT operators. You can  
 use parentheses to group clauses and force precedence. The AND operator has  
 precedence over other operators. For example: (LastName = 'Smith' OR LastName  
 = 'Jones') AND FirstName = 'John' When creating comparison expressions, the

1 following operators are allowed: < > <= >= <> = IN LIKE The following  
 2 arithmetic operators are also supported in expressions: + (addition) - (subtraction)  
 3 \* (multiplication) / (division) % (modulus) STRING OPERATORS To  
 4 concatenate a string, use the + character. Whether string comparisons are case-  
 5 sensitive or not is determined by the value of the **System.Data.DataSet** class's  
 6 **System.Data.DataSet.CaseSensitive** property. However, you can override that  
 7 value with the **System.Data.DataTable** class's  
 8 **System.Data.DataTable.CaseSensitive** property.

9       ExtendedProperties

10       ToString

11  
12 [C#] public PropertyCollection ExtendedProperties {get;}

13 [C++] public: \_\_property PropertyCollection\* get\_ExtendedProperties();

14 [VB] Public ReadOnly Property ExtendedProperties As PropertyCollection

15 [JScript] public function get ExtendedProperties() : PropertyCollection;

### 17 *Description*

18       Gets the collection of custom user information.

19       The **System.Data.DataColumn.ExtendedProperties** property allows you  
 20 to store custom information with the object. For example, you may store a time  
 21 when the data should be refreshed.

22       MaxLength

23       ToString

24  
25 [C#] public int MaxLength {get; set;}

```

1 [C++] public: __property int get_MaxLength();public: __property void
2 set_MaxLength(int);
3 [VB] Public Property MaxLength As Integer
4 [JScript] public function get MaxLength() : int;public function set
5 MaxLength(int);

```

#### *Description*

Gets or sets the maximum length of a text column.

The **System.Data.DataColumn.MaxLength** property is ignored for non-text columns.

Namespace

ToString

```

14 [C#] public string Namespace {get; set;}
15 [C++] public: __property String* get_Namespace();public: __property void
16 set_Namespace(String*);
17 [VB] Public Property Namespace As String
18 [JScript] public function get Namespace() : String;public function set
19 Namespace(String);

```

#### *Description*

Gets or sets the namespace of the **System.Data.DataColumn**.

The **System.Data.DataColumn.Namespace** property is used when reading and writing an XML document into a **System.Data.DataTable** in the **System.Data.DataSet** using the

1 **System.Data.DataSet.ReadXml(System.Xml.XmlReader) ,**  
 2 **System.Data.DataSet.WriteXml(System.IO.Stream) ,**  
 3 **System.Data.DataSet.ReadXmlSchema(System.Xml.XmlReader) , or**  
 4 **System.Data.DataSet.WriteXmlSchema(System.IO.Stream) methods.**

5 Ordinal

6 ToString

7  
 8 [C#] public int Ordinal {get;}  
 9 [C++] public: \_\_property int get\_Ordinal();  
 10 [VB] Public ReadOnly Property Ordinal As Integer  
 11 [JScript] public function get Ordinal() : int;

12  
 13 *Description*

14 Gets the position of the column in the  
 15 **System.Data.DataColumnCollection** collection.

16 Prefix

17 ToString

18  
 19 [C#] public string Prefix {get; set;}  
 20 [C++] public: \_\_property String\* get\_Prefix();public: \_\_property void  
 21 set\_Prefix(String\*);  
 22 [VB] Public Property Prefix As String  
 23 [JScript] public function get Prefix() : String;public function set Prefix(String);

24  
 25 *Description*

Gets or sets an XML prefix that aliases the namespace of the **System.Data.DataTable** .

The **System.Data.DataTable.Prefix** is used throughout an XML document to identify elements which belong to the **System.Data.DataSet** object's namespace (as set by the **System.Data.DataSet.Namespace** property).

ReadOnly

ToString

[C#] public bool ReadOnly {get; set;}

[C++] public: \_\_property bool get\_ReadOnly();public: \_\_property void set\_ReadOnly(bool);

[VB] Public Property ReadOnly As Boolean

[JScript] public function get ReadOnly() : Boolean;public function set ReadOnly(Boolean);

#### *Description*

Gets or sets a value indicating whether the column allows changes once a row has been added to the table.

Site

Table

ToString

#### *Description*

Gets the **System.Data.DataTable** to which the column belongs to.

Unique

ToString

[C#] public bool Unique {get; set;}

[C++] public: \_\_property bool get\_Unique();public: \_\_property void

set\_Unique(bool);

[VB] Public Property Unique As Boolean

[JScript] public function get Unique() : Boolean;public function set

Unique(Boolean);

### *Description*

Gets or sets a value indicating whether the values in each row of the column must be unique.

You can also add a **System.Data.UniqueConstraint** to the **System.Data.DataTable** to which this column belongs to ensure the values are unique.

CheckNotAllowNull

[C#] protected internal void CheckNotAllowNull();

[C++] protected public: void CheckNotAllowNull();

[VB] Protected Friend Dim Sub CheckNotAllowNull()

[JScript] package function CheckNotAllowNull();

### *Description*

CheckUnique

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

[C#] protected void CheckUnique();  
 [C++] protected: void CheckUnique();  
 [VB] Protected Sub CheckUnique()  
 [JScript] protected function CheckUnique();

*Description*

Throws an exception and the name of any column if its Unique property set to True and non-unique values are found in the column.

OnPropertyChanging

[C#] protected internal virtual void  
 OnPropertyChanging(PropertyChangedEventArgs pcevent);  
 [C++] protected public: virtual void  
 OnPropertyChanging(PropertyChangedEventArgs\* pcevent);  
 [VB] Overridable Protected Friend Dim Sub OnPropertyChanging(ByVal pcevent  
 As PropertyChangedEventArgs)  
 [JScript] package function OnPropertyChanging(pcevent :  
 PropertyChangedEventArgs);

*Description*

Raises the  
**System.Data.DataColumn.OnPropertyChanging(System.ComponentModel.P  
 ropertyChangedEventArgs) event.**

1 Raising an event invokes the event handler through a delegate. For an  
2 overview, see . A **System.ComponentModel.PropertyChangedEventArgs** that  
3 contains the event data.

#### 4 RaisePropertyChanging

5  
6 [C#] protected internal void RaisePropertyChanging(string name);  
7 [C++] protected public: void RaisePropertyChanging(String\* name);  
8 [VB] Protected Friend Dim Sub RaisePropertyChanging(ByVal name As String)  
9 [JScript] package function RaisePropertyChanging(name : String);  
10

#### 11 *Description*

12 Sends notification that the specified **System.Data.DataColumn** property is  
13 about to change. The name of the property that is about to change.

#### 14 ToString

15  
16 [C#] public override string ToString();  
17 [C++] public: String\* ToString();  
18 [VB] Overrides Public Function ToString() As String  
19 [JScript] public override function ToString() : String;  
20

#### 21 *Description*

22 Gets the **System.Data.DataColumn.Expression** of the column, if one  
23 exists.

24 *Return Value:* The **System.Data.DataColumn.Expression** value, if the property  
25 is set; otherwise, the **System.Data.DataColumn.ColumnName** property.

DataColumnChangeEventArgs class (System.Data)

ToString

### Description

Provides data for the **System.Data.DataTable.ColumnChanging** event.

The **System.Data.DataTable.ColumnChanging** event occurs when a change is made to a column's value in the **System.Data.DataTable** .

DataColumnChangeEventArgs

### Example Syntax:

ToString

[C#] public DataColumnChangeEventArgs(DataRow row, DataColumn column, object value);

[C++] public: DataColumnChangeEventArgs(DataRow\* row, DataColumn\* column, Object\* value);

[VB] Public Sub New(ByVal row As DataRow, ByVal column As DataColumn, ByVal value As Object)

[JScript] public function DataColumnChangeEventArgs(row : DataRow, column : DataColumn, value : Object);

### Description

Initializes a new instance of the **System.Data.DataColumnChangeEventArgs** class. The **System.Data.DataRow**

1 with the changing value. The **System.Data.DataColumn** with the changing value.

2 The new value.

3 Column

4 ToString

6 [C#] public DataColumn Column {get;}

7 [C++] public: \_\_property DataColumn\* get\_Column();

8 [VB] Public ReadOnly Property Column As DataColumn

9 [JScript] public function get Column() : DataColumn;

11 *Description*

12 Gets the **System.Data.DataColumn** with a changing value.

13 ProposedValue

14 ToString

16 [C#] public object ProposedValue {get; set;}

17 [C++] public: \_\_property Object\* get\_ProposedValue();public: \_\_property void

18 set\_ProposedValue(Object\*);

19 [VB] Public Property ProposedValue As Object

20 [JScript] public function get ProposedValue() : Object;public function set

21 ProposedValue(Object);

23 *Description*

24 Gets or sets the proposed value.

25 Row

ToString

[C#] public DataRow Row {get;}

[C++] public: \_\_property DataRow\* get\_Row();

[VB] Public ReadOnly Property Row As DataRow

[JScript] public function get Row() : DataRow;

### *Description*

Gets the **System.Data.DataRow** with a changing value.

**DataColumnChangeEventHandler** delegate (System.Data)

ToString

### *Description*

Represents the method that will handle the the

**System.Data.DataTable.ColumnChanging** event. The source of the event. A

**System.Data.DataColumnChangeEventArgs** that contains the event data.

When you create a **System.Data.DataColumnChangeEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, until you remove the delegate. For more information about delegates, see [Represents the method that will handle the the \*\*System.Data.DataTable.ColumnChanging\*\* event.](#)

**DataColumnCollection** class (System.Data)

ToString

### *Description*

Represents a collection of **System.Data.DataColumn** objects for a **System.Data.DataTable** .

The **System.Data.DataColumnCollection** defines the schema of a **System.Data.DataTable** , and determines what kind of data each **System.Data.DataColumn** can contain. You can access the **System.Data.DataColumnCollection** through the **System.Data.DataTable.Columns** property of the **System.Data.DataTable** object.

Count

IsReadOnly

IsSynchronized

Item

ToString

**System.Data.DataColumn**

### *Description*

Gets the **System.Data.DataColumn** from the collection at the specified index.

The **System.Data.DataColumnCollection.Contains(System.String)** method can be used to test for the existence of a column, which is useful before attempting to use **System.Data.DataColumnCollection.Item(System.Int32)** . The zero-based index of the column to return.

Item

ToString

[C#] public virtual DataColumn this[string name] {get;}

[C++] public: \_\_property virtual DataColumn\* get\_Item(String\* name);

[VB] Overridable Public Default ReadOnly Property Item(ByVal name As String)

As DataColumn

[JScript] returnValue = DataColumnCollectionObject.Item(name);

### *Description*

Gets the **System.Data.DataColumn** from the collection with the specified name.

**System.Data.DataColumnCollection.Item(System.Int32)** is not case-sensitive when searching for column names. The **System.Data.DataColumn.ColumnName** of the column to return.

List

ToString

[C#] protected override ArrayList List {get;}

[C++] protected: \_\_property virtual ArrayList\* get\_List();

[VB] Overrides Protected ReadOnly Property List As ArrayList

[JScript] protected function get List() : ArrayList;

### *Description*

Gets the list of the collection items.

SyncRoot

ToString

*Description*

Occurs when the columns collection changes, either by adding or removing a column.

The **System.Data.DataColumnCollection.Contains(System.String)** and **System.Data.DataColumnCollection.CanRemove(System.Data.DataColumn)** methods can be used to determine if a column exists and can be removed.

Add

[C#] public virtual DataColumn Add();

[C++] public: virtual DataColumn\* Add();

[VB] Overridable Public Function Add() As DataColumn

[JScript] public function Add() : DataColumn;

*Description*

Creates and adds a **System.Data.DataColumn** to a **System.Data.DataColumnCollection**.

*Return Value:* The newly created **System.Data.DataColumn**.

A default name ("Column1", "Column2", etc.) is given to the column.

Add

[C#] public void Add(DataColumn column);

[C++] public: void Add(DataColumn\* column);

[VB] Public Sub Add(ByVal column As DataColumn)

[JScript] public function Add(column : DataColumn); Adds a

**System.Data.DataColumn** to the **System.Data.DataColumnCollection** .

### *Description*

Adds the specified **System.Data.DataColumn** to the **System.Data.DataColumnCollection** .

If the collection is successfully changed by adding or removing columns, the **System.Data.DataColumnCollection.CollectionChanged** event occurs. The **System.Data.DataColumn** to add.

### *Add*

[C#] public virtual DataColumn Add(string columnName);

[C++] public: virtual DataColumn\* Add(String\* columnName); .

[VB] Overridable Public Function Add(ByVal columnName As String) As DataColumn

[JScript] public function Add(columnName : String) : DataColumn;

### *Description*

Creates and adds a **System.Data.DataColumn** with the specified name to the **System.Data.DataColumnCollection** .

*Return Value:* The newly created **System.Data.DataColumn** .

By default, the column's **System.Data.DataColumn.DataType** is string.  
The name of the column.

## Add

```
[C#] public virtual DataColumn Add(string columnName, Type type);  
[C++] public: virtual DataColumn* Add(String* columnName, Type* type);  
[VB] Overridable Public Function Add(ByVal columnName As String, ByVal  
type As Type) As DataColumn  
[JScript] public function Add(columnName : String, type : Type) : DataColumn;
```

### *Description*

Creates and adds a **System.Data.DataColumn** with the specified name and type to the **System.Data.DataColumnCollection**.

*Return Value:* The newly created **System.Data.DataColumn**.

If **null** or an empty string ("" ) is passed in for the name, a default name ("Column1", "Column2", etc.) is given to the column. The **System.Data.DataColumn.ColumnName** to create the column with. The column's **System.Data.DataColumn.DataType**.

## Add

```
[C#] public virtual DataColumn Add(string columnName, Type type, string  
expression);  
[C++] public: virtual DataColumn* Add(String* columnName, Type* type,  
String* expression);  
[VB] Overridable Public Function Add(ByVal columnName As String, ByVal  
type As Type, ByVal expression As String) As DataColumn  
[JScript] public function Add(columnName : String, type : Type, expression :
```

String) : DataColumn;

### *Description*

Creates and adds a **System.Data.DataColumn** with the specified name, type, and compute expression to the **System.Data.DataColumnCollection**.

*Return Value:* The newly created **System.Data.DataColumn**.

If **null** or an empty string ("" ) is passed in for the name, a default name ("Column1", "Column2", etc.) is given to the column. The column name. The **System.Data.DataColumn.DataType** of the column. The expression to assign to the **System.Data.DataColumn.Expression** property.

### **AddRange**

[C#] public void AddRange(DataColumn[] columns);

[C++] public: void AddRange(DataColumn\* columns[]);

[VB] Public Sub AddRange(ByVal columns() As DataColumn)

[JScript] public function AddRange(columns : DataColumn[]);

### *Description*

Copies the elements of the specified **System.Data.DataColumn** array to the end of the collection. The array of **System.Data.DataColumn** objects to add to the collection.

### **CanRemove**

[C#] public bool CanRemove(DataColumn column);

[C++] public: bool CanRemove(DataColumn\* column);

1 [VB] Public Function CanRemove(ByVal column As DataColumn) As Boolean  
 2 [JScript] public function CanRemove(column : DataColumn) : Boolean;

3  
 4 *Description*

5 Checks whether a given column can be removed from the collection.

6 *Return Value:* **true** if the column can be removed; otherwise, **false** .

7 The

8 **System.Data.DataColumnCollection.CanRemove(System.Data.DataColumn)**

9 method performs several checks before returning a **true** or **false** including the  
 10 following: whether the column exists, belongs to the table, or is involved in a  
 11 constraint or relation. A **System.Data.DataColumn** in the collection.

12 Clear

13  
 14 [C#] public void Clear();

15 [C++] public: void Clear();

16 [VB] Public Sub Clear()

17 [JScript] public function Clear();

18  
 19 *Description*

20 Clears the collection of any columns.

21 If the collection is successfully changed by adding or removing columns, the  
 22 **System.Data.DataColumnCollection.OnCollectionChanged(System.ComponentModel.CollectionChangeEventArgs)** event occurs.  
 23

24 Contains  
 25

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```

[C#] public bool Contains(string name);
[C++] public: bool Contains(String* name);
[VB] Public Function Contains(ByVal name As String) As Boolean
[JScript] public function Contains(name : String) : Boolean;
    
```

*Description*

Checks whether the collection contains a column with the specified name.  
*Return Value:* **true** if a column exists with this name; otherwise, **false** .

The **System.Data.DataColumnCollection.Contains(System.String)** method can confirm the existence of a column before performing further operations on the column. The **System.Data.DataColumn.ColumnName** of the column.

**IndexOf**

```

[C#] public virtual int IndexOf(DataColumn column);
[C++] public: virtual int IndexOf(DataColumn* column);
[VB] Overridable Public Function IndexOf(ByVal column As DataColumn) As Integer
[JScript] public function IndexOf(column : DataColumn) : int;
    
```

*Description*

Gets the index of a column specified by name.  
*Return Value:* The index of the column specified by *columnName* if it is found; otherwise, -1.

The  
**System.Data.DataColumnCollection.IndexOf(System.Data.DataColumn)**  
 method is not case-sensitive.

**IndexOf**

[C#] public int IndexOf(string columnName);  
 [C++] public: int IndexOf(String\* columnName);  
 [VB] Public Function IndexOf(ByVal columnName As String) As Integer  
 [JScript] public function IndexOf(columnName : String) : int; Returns the index of  
 a column specified by name.

**OnCollectionChanged**

[C#] protected virtual void OnCollectionChanged(CollectionChangeEventArgs  
 ccevent);  
 [C++] protected: virtual void OnCollectionChanged(CollectionChangeEventArgs\*  
 ccevent);  
 [VB] Overridable Protected Sub OnCollectionChanged(ByVal ccevent As  
 CollectionChangeEventArgs)  
 [JScript] protected function OnCollectionChanged(ccevent :  
 CollectionChangeEventArgs);

## *Description*

Raises the  
**System.Data.DataColumnCollection.OnCollectionChanged(System.Compone  
 ntModel.CollectionChangeEventArgs)** event.

1 Raising an event invokes the event handler through a delegate. For an  
2 overview, see . A **System.ComponentModel.CollectionChangeEventArgs** that  
3 contains the event data.

#### 4 OnCollectionChanging

5  
6 [C#] protected internal virtual void

7 OnCollectionChanging(CollectionChangeEventArgs ccevent);

8 [C++] protected public: virtual void

9 OnCollectionChanging(CollectionChangeEventArgs\* ccevent);

10 [VB] Overridable Protected Friend Dim Sub OnCollectionChanging(ByVal

11 ccevent As CollectionChangeEventArgs)

12 [JScript] package function OnCollectionChanging(ccevent :

13 CollectionChangeEventArgs);

#### 14 15 *Description*

16 Raises the

17 **System.Data.DataColumnCollection.OnCollectionChanging(System.Compon**  
18 **entModel.CollectionChangeEventArgs)** event.

19 Raising an event invokes the event handler through a delegate. For an  
20 overview, see . A **System.ComponentModel.CollectionChangeEventArgs** that  
21 contains the event data.

#### 22 Remove

23  
24 [C#] public void Remove(DataColumn column);

25 [C++] public: void Remove(DataColumn\* column);

[VB] Public Sub Remove(ByVal column As DataColumn)

[JScript] public function Remove(column : DataColumn); Removes a column from the collection.

#### *Description*

Removes the specified **System.Data.DataColumn** from the collection.

If the collection is successfully changed by adding or removing columns, the **System.Data.DataColumnCollection.OnCollectionChanged(System.ComponentModel.CollectionChangeEventArgs)** event occurs. The **System.Data.DataColumn** to remove.

Remove

[C#] public void Remove(string name);

[C++] public: void Remove(String\* name);

[VB] Public Sub Remove(ByVal name As String)

[JScript] public function Remove(name : String);

#### *Description*

Removes the column with the specified name from the collection.

If the collection is successfully changed by adding or removing columns, the **System.Data.DataColumnCollection.OnCollectionChanged(System.ComponentModel.CollectionChangeEventArgs)** event occurs. The name of the column to remove.

RemoveAt

1  
2 [C#] public void RemoveAt(int index);

3 [C++] public: void RemoveAt(int index);

4 [VB] Public Sub RemoveAt(ByVal index As Integer)

5 [JScript] public function RemoveAt(index : int);

6  
7 *Description*

8 Removes the column at the specified index from the collection.

9 If the collection is successfully changed by adding or removing columns, the  
10 **System.Data.DataColumnCollection.OnCollectionChanged(System.Compone**  
11 **ntModel.CollectionChangeEventArgs)** event occurs. The index of the column to  
12 remove.

13 DataException class (System.Data)

14 ToString

15  
16  
17 *Description*

18 Represents the exception that is thrown when errors are generated using  
19 ADO.NET components.

20 DataException

21 *Example Syntax:*

22 ToString

23  
24 [C#] public DataException();

25 [C++] public: DataException();

1 [VB] Public Sub New()  
2 [JScript] public function DataException();

3  
4 *Description*

5       Initializes a new instance of the **System.Data.DataException** class.

6       DataException

7       *Example Syntax:*

8       ToString

9  
10 [C#] public DataException(string s);  
11 [C++] public: DataException(String\* s);  
12 [VB] Public Sub New(ByVal s As String)  
13 [JScript] public function DataException(s : String);

14  
15 *Description*

16       Initializes a new instance of the **System.Data.DataException** class with  
17 the specified string. The string to display when the exception is thrown.

18       DataException

19       *Example Syntax:*

20       ToString

21  
22 [C#] public DataException(SerializationInfo info, StreamingContext context);  
23 [C++] public: DataException(SerializationInfo\* info, StreamingContext context);  
24 [VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As  
25 StreamingContext)

[JScript] public function DataException(info : SerializationInfo, context : StreamingContext); Initializes a new instance of the **System.Data.DataException** class.

#### *Description*

Initializes a new instance of the **System.Data.DataException** class. The data necessary to serialize or deserialize an object. Description of the source and destination of the specified serialized stream.

DataException

*Example Syntax:*

ToString

[C#] public DataException(string s, Exception innerException);

[C++] public: DataException(String\* s, Exception\* innerException);

[VB] Public Sub New(ByVal s As String, ByVal innerException As Exception)

[JScript] public function DataException(s : String, innerException : Exception);

Initializes a new instance of the **System.Data.DataException** class.

#### *Description*

Initializes a new instance of the **System.Data.DataException** class with the specified string and inner exception.

You can create a new exception that catches an earlier exception. The code that handles the second exception can make use of the additional information from the earlier exception, also called an inner exception, to examine the cause of the

1 initial error. The string to display when the exception is thrown. A reference to an  
2 inner exception.

3 HelpLink

4 HResult

5 InnerException

6 Message

7 Source

8 StackTrace

9 TargetSite

10 DataRelation class (System.Data)

11 ToString

12  
13  
14 *Description*

15 Represents a parent/child relationship between two  
16 **System.Data.DataTable** objects.

17 A **System.Data.DataRelation** is used to relate two  
18 **System.Data.DataTable** objects to each other through  
19 **System.Data.DataColumn** objects. For example, in a Customer/Orders  
20 relationship, the Customers table is the parent and the Orders table is the child of  
21 the relationship. This is similar to a primary key/foreign key relationship. For  
22 more information, see .

23 DataRelation

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public DataRelation(string relationName, DataColumn parentColumn,
3 DataColumn childColumn);
4 [C++] public: DataRelation(String* relationName, DataColumn* parentColumn,
5 DataColumn* childColumn);
6 [VB] Public Sub New(ByVal relationName As String, ByVal parentColumn As
7 DataColumn, ByVal childColumn As DataColumn)
8 [JScript] public function DataRelation(relationName : String, parentColumn :
9 DataColumn, childColumn : DataColumn); Initializes a new instance of the
10 System.Data.DataRelation class.
11

```

### *Description*

Initializes a new instance of the **System.Data.DataRelation** class using the specified **System.Data.DataRelation** name, and parent and child **System.Data.DataColumn** objects. The name of the **System.Data.DataRelation** . If **null** or an empty string (""), a default name will be given when the created object is added to the **System.Data.DataRelationCollection** . The parent **System.Data.DataColumn** in the relationship. The child **System.Data.DataColumn** in the relationship.

DataRelation

*Example Syntax:*

ToString

```

23
24 [C#] public DataRelation(string relationName, DataColumn[] parentColumns,
25 DataColumn[] childColumns);

```

```

1 [C++] public: DataRelation(String* relationName, DataColumn*
2 parentColumns[], DataColumn* childColumns[]);
3 [VB] Public Sub New(ByVal relationName As String, ByVal parentColumns() As
4 DataColumn, ByVal childColumns() As DataColumn)
5 [JScript] public function DataRelation(relationName : String, parentColumns :
6 DataColumn[], childColumns : DataColumn[]);
7

```

### *Description*

Initializes a new instance of the **System.Data.DataRelation** class using the specified **System.Data.DataRelation** name and matched arrays of parent and child **System.Data.DataColumn** objects. The name of the relation. If **null** or an empty string (""), a default name will be given when the created object is added to the **System.Data.DataRelationCollection** . An array of parent **System.Data.DataColumn** objects. An array of child **System.Data.DataColumn** objects.

**DataRelation**

*Example Syntax:*

**ToString**

```

20 [C#] public DataRelation(string relationName, DataColumn parentColumn,
21 DataColumn childColumn, bool createConstraints);
22 [C++] public: DataRelation(String* relationName, DataColumn* parentColumn,
23 DataColumn* childColumn, bool createConstraints);
24 [VB] Public Sub New(ByVal relationName As String, ByVal parentColumn As
25 DataColumn, ByVal childColumn As DataColumn, ByVal createConstraints As

```

Boolean)

[JScript] public function DataRelation(relationName : String, parentColumn : DataColumn, childColumn : DataColumn, createConstraints : Boolean);

### *Description*

Initializes a new instance of the **System.Data.DataRelation** class using the specified name, parent and child **System.Data.DataColumn** objects, and a value indicating whether to create constraints. The name of the relation. If **null** or an empty string (""), a default name will be given when the created object is added to the **System.Data.DataRelationCollection** . The parent **System.Data.DataColumn** in the relation. The child **System.Data.DataColumn** in the relation. A value indicating whether constraints are created.

**DataRelation**

*Example Syntax:*

**ToString**

[C#] public DataRelation(string relationName, DataColumn[] parentColumns, DataColumn[] childColumns, bool createConstraints);

[C++] public: DataRelation(String\* relationName, DataColumn\* parentColumns[], DataColumn\* childColumns[], bool createConstraints);

[VB] Public Sub New(ByVal relationName As String, ByVal parentColumns() As DataColumn, ByVal childColumns() As DataColumn, ByVal createConstraints As Boolean)

[JScript] public function DataRelation(relationName : String, parentColumns : DataColumn[], childColumns : DataColumn[], createConstraints : Boolean);

## Description

Initializes a new instance of the **System.Data.DataRelation** class using the specified name, matched arrays of parent and child **System.Data.DataColumn** objects, and value indicating whether to create constraints. The name of the relation. If **null** or an empty string (""), a default name will be given when the created object is added to the **System.Data.DataRelationCollection**. An array of parent **System.Data.DataColumn** objects. An array of child **System.Data.DataColumn** objects. A value indicating whether to create constraints.

**DataRelation**

*Example Syntax:*

**ToString**

```
[C#] public DataRelation(string relationName, string parentTableName, string  
childTableName, string[] parentColumnNames, string[] childColumnNames, bool  
nested);
```

```
[C++] public: DataRelation(String* relationName, String* parentTableName,  
String* childTableName, String* parentColumnNames __gc[], String*  
childColumnNames __gc[], bool nested);
```

```
[VB] Public Sub New(ByVal relationName As String, ByVal parentTableName  
As String, ByVal childTableName As String, ByVal parentColumnNames() As  
String, ByVal childColumnNames() As String, ByVal nested As Boolean)
```

```
[JScript] public function DataRelation(relationName : String, parentTableName :  
String, childTableName : String, parentColumnNames : String[],
```

childColumnNames : String[], nested : Boolean);

### Description

Initializes a new instance of the **System.Data.DataRelation** class using the specified **System.Data.DataRelation** name, parent and child

**System.Data.DataTable** names, a matching array of parent and child

**System.Data.DataColumn** objects, and a value indicating whether relationships are nested. The name of the relation. If **null** or an empty string (""), a default name will be given when the created object is added to the

**System.Data.DataRelationCollection** . The name of the

**System.Data.DataTable** that is the parent table of the relation. The name of the

**System.Data.DataTable** that is the child table of the relation. An array of

**System.Data.DataColumn** object names in the parent **System.Data.DataTable**

of the relation. An array of **System.Data.DataColumn** object names in the child

**System.Data.DataTable** of the relation. A value indicating whether relationships are nested.

ChildColumns

ToString

[C#] public virtual DataColumn[] ChildColumns {get;}

[C++] public: \_\_property virtual DataColumn\* get\_ChildColumns();

[VB] Overridable Public ReadOnly Property ChildColumns As DataColumn ()

[JScript] public function get ChildColumns() : DataColumn[];

### Description

1 Gets the child **System.Data.DataColumn** objects of this relation.  
 2 **ChildKeyConstraint**  
 3 **ToString**  
 4  
 5 [C#] public virtual ForeignKeyConstraint ChildKeyConstraint {get;}  
 6 [C++] public: \_\_property virtual ForeignKeyConstraint\*  
 7 get\_ChildKeyConstraint();  
 8 [VB] Overridable Public ReadOnly Property ChildKeyConstraint As  
 9 ForeignKeyConstraint  
 10 [JScript] public function get ChildKeyConstraint() : ForeignKeyConstraint;  
 11

12 *Description*

13 Gets the **System.Data.ForeignKeyConstraint** for the relation.  
 14 **ChildTable**  
 15 **ToString**  
 16

17 [C#] public virtual DataTable ChildTable {get;}  
 18 [C++] public: \_\_property virtual DataTable\* get\_ChildTable();  
 19 [VB] Overridable Public ReadOnly Property ChildTable As DataTable  
 20 [JScript] public function get ChildTable() : DataTable;  
 21

22 *Description*

23 Gets the child table of this relation.  
 24 **DataSet**  
 25 **ToString**

```

1
2 [C#] public virtual DataSet DataSet {get;}
3 [C++] public: __property virtual DataSet* get_DataSet();
4 [VB] Overridable Public ReadOnly Property DataSet As DataSet
5 [JScript] public function get DataSet() : DataSet;
6

```

### *Description*

Gets the **System.Data.DataSet** to which the **System.Data.DataRelation** belongs.

The **System.Data.DataRelationCollection** associated with a **System.Data.DataSet** is accessed through the **System.Data.DataSet.Relations** property of the **System.Data.DataSet** object.

ExtendedProperties

ToString

```

13
14
15
16 [C#] public PropertyCollection ExtendedProperties {get;}
17 [C++] public: __property PropertyCollection* get_ExtendedProperties();
18 [VB] Public ReadOnly Property ExtendedProperties As PropertyCollection
19 [JScript] public function get ExtendedProperties() : PropertyCollection;
20

```

### *Description*

Gets the collection that stores customized properties.

Nested

ToString

[C#] public virtual bool Nested {get; set;}

[C++] public: \_\_property virtual bool get\_Nested();public: \_\_property virtual void set\_Nested(bool);

[VB] Overridable Public Property Nested As Boolean

[JScript] public function get Nested() : Boolean;public function set Nested(Boolean);

### *Description*

Gets or sets a value indicating whether **System.Data.DataRelation** objects are nested.

You can use **System.Data.DataRelation** objects to define hierarchical relationships, such as those specified in XML. For more information, see .

ParentColumns

ToString

[C#] public virtual DataColumn[] ParentColumns {get;}

[C++] public: \_\_property virtual DataColumn\* get\_ParentColumns();

[VB] Overridable Public ReadOnly Property ParentColumns As DataColumn ()

[JScript] public function get ParentColumns() : DataColumn[];

### *Description*

Gets an array of **System.Data.DataColumn** objects that are the parent columns of this **System.Data.DataRelation** .

ParentKeyConstraint

1	ToString
2	
3	[C#] public virtual UniqueConstraint ParentKeyConstraint {get;}
4	[C++] public: __property virtual UniqueConstraint* get_ParentKeyConstraint();
5	[VB] Overridable Public ReadOnly Property ParentKeyConstraint As
6	UniqueConstraint
7	[JScript] public function get ParentKeyConstraint() : UniqueConstraint;
8	
9	<i>Description</i>
10	Gets the <b>System.Data.UniqueConstraint</b> that ensures values in the parent
11	column of a <b>System.Data.DataRelation</b> are unique.
12	ParentTable
13	ToString
14	
15	[C#] public virtual DataTable ParentTable {get;}
16	[C++] public: __property virtual DataTable* get_ParentTable();
17	[VB] Overridable Public ReadOnly Property ParentTable As DataTable
18	[JScript] public function get ParentTable() : DataTable;
19	
20	<i>Description</i>
21	Gets the parent <b>System.Data.DataTable</b> of this
22	<b>System.Data.DataRelation</b> .
23	RelationName
24	ToString
25	

```

[C#] public virtual string RelationName {get; set;}
[C++] public: __property virtual String* get_RelationName();public: __property
virtual void set_RelationName(String*);
[VB] Overridable Public Property RelationName As String
[JScript] public function get RelationName() : String;public function set
RelationName(String);
    
```

#### *Description*

Gets or sets the name used to retrieve a **System.Data.DataRelation** from the **System.Data.DataRelationCollection** .

Use the **System.Data.DataRelation.RelationName** property to retrieve a **System.Data.DataRelation** from the **System.Data.DataRelationCollection** .

#### *CheckStateForProperty*

```

[C#] protected void CheckStateForProperty();
[C++] protected: void CheckStateForProperty();
[VB] Protected Sub CheckStateForProperty()
[JScript] protected function CheckStateForProperty();
    
```

#### *Description*

Ensures that the **System.Data.DataRelation** is a valid object.

**System.Data.DataRelation.CheckStateForProperty** verifies the validity of a **System.Data.DataRelation** object, even if it does not belong to a **System.Data.DataSet** .

## OnPropertyChanging

[C#] protected internal void OnPropertyChanging(PropertyChangedEventArgs pcevent);

[C++] protected public: void OnPropertyChanging(PropertyChangedEventArgs\* pcevent);

[VB] Protected Friend Dim Sub OnPropertyChanging(ByVal pcevent As PropertyChangedEventArgs)

[JScript] package function OnPropertyChanging(pcevent : PropertyChangedEventArgs);

### *Description*

## RaisePropertyChanging

[C#] protected internal void RaisePropertyChanging(string name);

[C++] protected public: void RaisePropertyChanging(String\* name);

[VB] Protected Friend Dim Sub RaisePropertyChanging(ByVal name As String)

[JScript] package function RaisePropertyChanging(name : String);

### *Description*

## ToString

[C#] public override string ToString();

1 [C++] public: String\* ToString();  
 2 [VB] Overrides Public Function ToString() As String  
 3 [JScript] public override function ToString() : String;

4  
 5 *Description*

6 Gets the **System.Data.DataRelation.RelationName** , if one exists.

7 *Return Value:* The value of the **System.Data.DataRelation.RelationName**  
 8 property.

9 DataRelationCollection class (System.Data)

10 ToString

11  
 12  
 13 *Description*

14 Represents the collection of **System.Data.DataRelation** objects for this  
 15 **System.Data.DataSet** .

16 A **System.Data.DataRelationCollection** object enables navigation  
 17 between related parent/child **System.Data.DataTable** objects.

18 DataRelationCollection

19 *Example Syntax:*

20 ToString

21  
 22 [C#] protected DataRelationCollection();

23 [C++] protected: DataRelationCollection();

24 [VB] Protected Sub New()

25 [JScript] protected function DataRelationCollection();

1 Count  
2 IsReadOnly  
3 IsSynchronized  
4 Item  
5 ToString

6  
7  
8 *Description*

9 Gets the **System.Data.DataRelation** object specified by name. The name  
10 of the relation to find.

11 Item  
12 ToString

13  
14 [C#] public abstract DataRelation this[int index] {get;}

15 [C++] public: \_\_property virtual DataRelation\* get\_Item(int index) = 0;

16 [VB] MustOverride Public Default ReadOnly Property Item(ByVal index As  
17 Integer) As DataRelation

18 [JScript] abstract returnValue = DataRelationCollectionObject.Item(index); Gets  
19 the specified **System.Data.DataRelation** from the collection.

20  
21 *Description*

22 Gets the **System.Data.DataRelation** object at the specified index. The  
23 zero-based index to find.

24 List  
25 SyncRoot

ToString

*Description*

Occurs when the collection has changed.

Add

[C#] public void Add(DataRelation relation);

[C++] public: void Add(DataRelation\* relation);

[VB] Public Sub Add(ByVal relation As DataRelation)

[JScript] public function Add(relation : DataRelation); Adds a

**System.Data.DataRelation** to the **System.Data.DataRelationCollection** .

*Description*

Adds a **System.Data.DataRelation** to the

**System.Data.DataRelationCollection** .

If the relation is successfully added to the collection, the

**System.Data.DataRelationCollection.CollectionChanged** event fires. The

**DataRelation** to add to the collection.

Add

[C#] public virtual DataRelation Add(DataColumn parentColumn, DataColumn  
childColumn);

[C++] public: virtual DataRelation\* Add(DataColumn\* parentColumn,  
DataColumn\* childColumn);

```

1 [VB] Overridable Public Function Add(ByVal parentColumn As DataColumn,
2   ByVal childColumn As DataColumn) As DataRelation
3 [JScript] public function Add(parentColumn : DataColumn, childColumn :
4   DataColumn) : DataRelation;

```

#### *Description*

Creates a relation given the parameters and adds it to the collection. The name is defaulted. An ArgumentException is generated if this relation already belongs to this collection or belongs to another collection. An InvalidConstraintException is generated if the relation can't be created based on the parameters. The CollectionChanged event is fired if it succeeds.

*Return Value:* The created relation. parent column of relation. child column of relation.

#### *Add*

```

16 [C#] public virtual DataRelation Add(DataColumn[] parentColumns,
17   DataColumn[] childColumns);
18 [C++] public: virtual DataRelation* Add(DataColumn* parentColumns[],
19   DataColumn* childColumns[]);
20 [VB] Overridable Public Function Add(ByVal parentColumns() As DataColumn,
21   ByVal childColumns() As DataColumn) As DataRelation
22 [JScript] public function Add(parentColumns : DataColumn[], childColumns :
23   DataColumn[]) : DataRelation;

```

#### *Description*

Creates a relation given the parameters and adds it to the collection. The name is defaulted. An `ArgumentException` is generated if this relation already belongs to this collection or belongs to another collection. An `InvalidConstraintException` is generated if the relation can't be created based on the parameters. The `CollectionChanged` event is fired if it succeeds.

*Return Value:* The created relation. parent columns of relation. child columns of relation.

#### Add

[C#] public virtual `DataRelation` Add(string name, `DataColumn` parentColumn, `DataColumn` childColumn);

[C++] public: virtual `DataRelation*` Add(String\* name, `DataColumn*` parentColumn, `DataColumn*` childColumn);

[VB] Overridable Public Function Add(ByVal name As String, ByVal parentColumn As `DataColumn`, ByVal childColumn As `DataColumn`) As `DataRelation`

[JScript] public function Add(name : String, parentColumn : `DataColumn`, childColumn : `DataColumn`) : `DataRelation`;

#### Description

Creates a relation given the parameters and adds it to the collection. An `ArgumentException` is generated if this relation already belongs to this collection or belongs to another collection. A `DuplicateNameException` is generated if this collection already has a relation with the same name (case insensitive). An `InvalidConstraintException` is generated if the relation can't be created based on

the parameters. The **CollectionChanged** event is fired if it succeeds.

*Return Value:* The created relation. The name of the relation. parent column of relation. child column of relation.

#### Add

[C#] public virtual **DataRelation** Add(string name, **DataColumn**[] parentColumns, **DataColumn**[] childColumns);

[C++] public: virtual **DataRelation\*** Add(String\* name, **DataColumn\*** parentColumns[], **DataColumn\*** childColumns[]);

[VB] Overridable Public Function Add(ByVal name As String, ByVal parentColumns() As **DataColumn**, ByVal childColumns() As **DataColumn**) As **DataRelation**

[JScript] public function Add(name : String, parentColumns : **DataColumn**[], childColumns : **DataColumn**[]) : **DataRelation**;

#### Description

Creates a **System.Data.DataRelation** with the specified name, and arrays of parent and child columns, and adds it to the collection.

*Return Value:* The created **DataRelation** .

If the relation is successfully added to the collection, the **System.Data.DataRelationCollection.CollectionChanged** event fires. The name of the **DataRelation** to create. An array of parent **System.Data.DataColumn** objects. An array of child **DataColumn** objects.

#### Add

```

1
2 [C#] public virtual DataRelation Add(string name, DataColumn parentColumn,
3 DataColumn childColumn, bool createConstraints);
4 [C++] public: virtual DataRelation* Add(String* name, DataColumn*
5 parentColumn, DataColumn* childColumn, bool createConstraints);
6 [VB] Overridable Public Function Add(ByVal name As String, ByVal
7 parentColumn As DataColumn, ByVal childColumn As DataColumn, ByVal
8 createConstraints As Boolean) As DataRelation
9 [JScript] public function Add(name : String, parentColumn : DataColumn,
10 childColumn : DataColumn, createConstraints : Boolean) : DataRelation;
11

```

## 12 *Description*

13 Creates a relation given the parameters and adds it to the collection. An  
14 ArgumentException is generated if this relation already belongs to this collection  
15 or belongs to another collection. A DuplicateNameException is generated if this  
16 collection already has a relation with the same name (case insensitive). An  
17 InvalidConstraintException is generated if the relation can't be created based on  
18 the parameters. The CollectionChanged event is fired if it succeeds.

19 *Return Value:* The created relation. The name of the relation. parent column of  
20 relation. child column of relation. whether to create a constraints

## 21 *Add*

```

22
23 [C#] public virtual DataRelation Add(string name, DataColumn[] parentColumns,
24 DataColumn[] childColumns, bool createConstraints);
25 [C++] public: virtual DataRelation* Add(String* name, DataColumn*

```

```

parentColumns[], DataColumn* childColumns[], bool createConstraints);
[VB] Overridable Public Function Add(ByVal name As String, ByVal
parentColumns() As DataColumn, ByVal childColumns() As DataColumn, ByVal
createConstraints As Boolean) As DataRelation
[JScript] public function Add(name : String, parentColumns : DataColumn[],
childColumns : DataColumn[], createConstraints : Boolean) : DataRelation;
    
```

### *Description*

Creates a **System.Data.DataRelation** with the specified name, arrays of parent and child columns, and value specifying whether to create a constraint, and adds it to the collection.

**Return Value:** The created relation. The name of the **DataRelation** to create. An array of parent **System.Data.DataColumn** objects. An array of child **DataColumn** objects. **true** to create a constraint; otherwise **false** .

### **AddCore**

```

[C#] protected virtual void AddCore(DataRelation relation);
[C++] protected: virtual void AddCore(DataRelation* relation);
[VB] Overridable Protected Sub AddCore(ByVal relation As DataRelation)
[JScript] protected function AddCore(relation : DataRelation);
    
```

### *Description*

Performs verification on the table. An **ArgumentNullException** is generated if this relation is null. An **ArgumentException** is generated if this relation already belongs to this collection, belongs to another collection. A

DuplicateNameException is generated if this collection already has a relation with the same name (case insensitive). The relation to check.

### AddRange

[C#] public virtual void AddRange(DataRelation[] relations);

[C++] public: virtual void AddRange(DataRelation\* relations[]);

[VB] Overridable Public Sub AddRange(ByVal relations() As DataRelation)

[JScript] public function AddRange(relations : DataRelation[]);

### Description

Copies the elements of the specified **System.Data.DataRelation** array to the end of the collection. The array of **System.Data.DataRelation** objects to add to the collection.

### CanRemove

[C#] public virtual bool CanRemove(DataRelation relation);

[C++] public: virtual bool CanRemove(DataRelation\* relation);

[VB] Overridable Public Function CanRemove(ByVal relation As DataRelation)

As Boolean

[JScript] public function CanRemove(relation : DataRelation) : Boolean; Verifies if a given relation can be removed from the collection.

### Clear

[C#] public virtual void Clear();

[C++] public: virtual void Clear();

1 [VB] Overridable Public Sub Clear()

2 [JScript] public function Clear();

3

4 *Description*

5 Clears the collection of any relations.

6 Contains

7

8 [C#] public virtual bool Contains(string name);

9 [C++] public: virtual bool Contains(String\* name);

10 [VB] Overridable Public Function Contains(ByVal name As String) As Boolean

11 [JScript] public function Contains(name : String) : Boolean;

12

13 *Description*

14 Gets a value of true if this collection has a relation with the given name

15 (case insensitive), false otherwise.

16 *Return Value:* Whether a relation exists with this name. name to test.

17 GetDataSet

18

19 [C#] protected abstract DataSet GetDataSet();

20 [C++] protected: virtual DataSet\* GetDataSet() = 0;

21 [VB] MustOverride Protected Function GetDataSet() As DataSet

22 [JScript] protected abstract function GetDataSet() : DataSet;

23

24 *Description*

25

Gets the dataset for this collection.

*Return Value:* The dataSet.

IndexOf

[C#] public virtual int IndexOf(DataRelation relation);

[C++] public: virtual int IndexOf(DataRelation\* relation);

[VB] Overridable Public Function IndexOf(ByVal relation As DataRelation) As

Integer

[JScript] public function IndexOf(relation : DataRelation) : int; Returns the index of a specified **System.Data.DataRelation** .

IndexOf

[C#] public virtual int IndexOf(string relationName);

[C++] public: virtual int IndexOf(String\* relationName);

[VB] Overridable Public Function IndexOf(ByVal relationName As String) As

Integer

[JScript] public function IndexOf(relationName : String) : int; Returns the index of the relation with the given name (case insensitive), or -1 if the relation doesn't exist in the collection.

OnCollectionChanged

[C#] protected virtual void OnCollectionChanged(CollectionChangeEventArgs ccevent);

[C++] protected: virtual void OnCollectionChanged(CollectionChangeEventArgs\* ccevent);

[VB] Overridable Protected Sub OnCollectionChanged(ByVal ccevent As  
CollectionChangeEventArgs)

[JScript] protected function OnCollectionChanged(ccevent :  
CollectionChangeEventArgs);

### *Description*

Raises the  
**System.Data.DataRelationCollection.OnCollectionChanged(System.ComponentModel.CollectionChangeEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an  
overview, see . A **System.ComponentModel.CollectionChangeEventArgs** that  
contains the event data.

### **OnCollectionChanging**

[C#] protected internal virtual void  
OnCollectionChanging(CollectionChangeEventArgs ccevent);  
[C++] protected public: virtual void  
OnCollectionChanging(CollectionChangeEventArgs\* ccevent);

[VB] Overridable Protected Friend Dim Sub OnCollectionChanging(ByVal  
ccevent As CollectionChangeEventArgs)  
[JScript] package function OnCollectionChanging(ccevent :  
CollectionChangeEventArgs);

### *Description*

1       Raises the

2       **System.Data.DataRelationCollection.OnCollectionChanging(System.Compon**  
3       **entModel.CollectionChangeEventArgs)** event.

4       Raising an event invokes the event handler through a delegate. For an  
5       overview, see . A **System.ComponentModel.CollectionChangeEventArgs** that  
6       contains the event data.

7       Remove

8  
9       [C#] public void Remove(DataRelation relation);

10      [C++] public: void Remove(DataRelation\* relation);

11      [VB] Public Sub Remove(ByVal relation As DataRelation)

12      [JScript] public function Remove(relation : DataRelation); Removes the specified  
13      relation from the collection.

14  
15      *Description*

16      Removes the specified relation from the collection. An  
17      ArgumentNullException is generated if this relation is null. An  
18      ArgumentException is generated if this relation doesn't belong to this collection.  
19      The CollectionChanged event is fired if it succeeds. The relation to remove.

20      Remove

21  
22      [C#] public void Remove(string name);

23      [C++] public: void Remove(String\* name);

24      [VB] Public Sub Remove(ByVal name As String)

25      [JScript] public function Remove(name : String);

## *Description*

Removes the relation with the specified name from the collection. An `IndexOutOfRangeException` is generated if this collection doesn't have a relation with that name. The `CollectionChanged` event is fired if it succeeds. The name of the relation to remove.

### **RemoveAt**

[C#] `public void RemoveAt(int index);`

[C++] `public: void RemoveAt(int index);`

[VB] `Public Sub RemoveAt(ByVal index As Integer)`

[JScript] `public function RemoveAt(index : int);`

## *Description*

Removes the relation at the specified index from the collection. An `IndexOutOfRangeException` is generated if this collection doesn't have a relation at this index. The `CollectionChanged` event is fired if it succeeds. The index at which to remove a relation.

### **RemoveCore**

[C#] `protected virtual void RemoveCore(DataRelation relation);`

[C++] `protected: virtual void RemoveCore(DataRelation* relation);`

[VB] `Overridable Protected Sub RemoveCore(ByVal relation As DataRelation)`

[JScript] `protected function RemoveCore(relation : DataRelation);`

1  
2 *Description*

3 Does verification on the relation. An `ArgumentNullException` is generated  
4 if this relation is null. An `ArgumentException` is generated if this relation doesn't  
5 belong to this collection. The relation to check.

6 `DataRow` class (`System.Data`)

7 `ToString`

8  
9  
10 *Description*

11 Represents a row of data in a `System.Data.DataTable` .

12 The `System.Data.DataRow` and `System.Data.DataColumn` objects are  
13 primary components of a `System.Data.DataTable` . Use the  
14 `System.Data.DataRow` object and its properties and methods to retrieve and  
15 evaluate; and insert, delete, and update the values in the `System.Data.DataTable` .

16 The `System.Data.DataRowCollection` represents the actual  
17 `System.Data.DataRow` objects in the `System.Data.DataTable` , and the  
18 `System.Data.DataColumnCollection` contains the `System.Data.DataColumn`  
19 objects that describe the schema of the `System.Data.DataTable` . Use the  
20 overloaded `System.Data.DataRow.Item(System.Int32)` property to return or sets  
21 the value of a `System.Data.DataColumn` .

22 `DataRow`

23 *Example Syntax:*

24 `ToString`

```

1
2 [C#] protected internal DataRow(DataRowBuilder builder);
3 [C++] internal: DataRow(DataRowBuilder* builder);
4 [VB] Protected Friend Sub New(ByVal builder As DataRowBuilder)
5 [JScript] package function DataRow(builder : DataRowBuilder);
6

```

### *Description*

Initializes a new instance of the DataRow. builder

HasErrors

ToString

```

11
12 [C#] public bool HasErrors {get;}
13 [C++] public: __property bool get_HasErrors();
14 [VB] Public ReadOnly Property HasErrors As Boolean
15 [JScript] public function get HasErrors() : Boolean;
16

```

### *Description*

Gets a value indicating whether there are errors in a columns collection.

When validating data, you can set an error on any column in a row. Such a column, when displayed in the **System.Windows.Forms.DataGrid** control, is marked with a red exclamation point to signal the user that the column is in error.

Item

ToString

```

22
23
24
25 [C#] public object this[string columnName] {get; set;}

```

```

1  [C++] public: __property Object* get_Item(String* columnName);public:
2  __property void set_Item(String* columnName, Object*);
3  [VB] Public Default Property Item(ByVal columnName As String) As Object
4  [JScript] returnValue =
5  DataRowObject.Item(columnName);DataRowObject.Item(columnName) =
6  returnValue;
7

```

### *Description*

Gets or sets the data stored in the column specified by name.

When setting the property, an exception is generated if an exception occurs in the **System.Data.DataTable.ColumnChanging** event. The name of the column.

Item

ToString

```

16 [C#] public object this[DataColumn column] {get; set;}
17 [C++] public: __property Object* get_Item(DataColumn* column);public:
18 __property void set_Item(DataColumn* column, Object*);
19 [VB] Public Default Property Item(ByVal column As DataColumn) As Object
20 [JScript] returnValue =
21 DataRowObject.Item(column);DataRowObject.Item(column) = returnValue;
22

```

### *Description*

Gets or sets the data stored in the specified **System.Data.DataColumn**.

When setting the property, an exception is generated if an exception occurs in the **System.Data.DataTable.ColumnChanging** event. A **System.Data.DataColumn** that contains the data.

Item

ToString

[C#] public object this[int columnIndex] {get; set;}

[C++] public: \_\_property Object\* get\_Item(int columnIndex); public: \_\_property void set\_Item(int columnIndex, Object\*);

[VB] Public Default Property Item(ByVal columnIndex As Integer) As Object

[JScript] returnValue =

DataRowObject.Item(columnIndex); DataRowObject.Item(columnIndex) =

returnValue; Gets or sets data stored in a specified column.

### *Description*

Gets or sets the data stored in the column specified by index.

When setting the property, an exception is generated if an exception occurs in the **System.Data.DataTable.ColumnChanging** event. The zero-based index of the column

Item

ToString

[C#] public object this[string columnName, DataRowVersion version] {get;}

[C++] public: \_\_property Object\* get\_Item(String\* columnName, DataRowVersion version);

```

1 [VB] Public Default ReadOnly Property Item(ByVal columnName As String,
2   ByVal version As DataRowVersion) As Object
3 [JScript] returnValue = DataRowObject.Item(columnName, version);

```

#### *Description*

Gets the specified version of data stored in the named column.

The *version* shouldn't be confused with the

**System.Data.DataRow.RowState** property. The *version* argument describes the state of the data contained by the column in relation to the column's original value.

The **System.Data.DataRow.RowState** property describes the state of the entire row in relation to its parent **System.Data.DataTable**. The name of the column.

One of the **System.Data.DataRowVersion** values that specifies the desired row version. Possible values are **Default**, **Original**, **Current**, and **Proposed**.

Item

ToString

```

17 [C#] public object this[DataColumn column, DataRowVersion version] {get;}
18 [C++] public: __property Object* get_Item(DataColumn* column,
19   DataRowVersion version);
20 [VB] Public Default ReadOnly Property Item(ByVal column As DataColumn,
21   ByVal version As DataRowVersion) As Object
22 [JScript] returnValue = DataRowObject.Item(column, version);

```

#### *Description*

Gets the specified version of data stored in the specified  
**System.Data.DataColumn** .

The *version* shouldn't be confused with the  
**System.Data.DataRow.RowState** property. The *version* argument describes the  
state of the data contained by the column in relation to the column's original value.  
A **System.Data.DataColumn** that contains information about the column. One of  
the **System.Data.DataRowVersion** values that specifies the desired row version.  
Possible values are **Default**, **Original**, **Current**, and **Proposed**.

Item

ToString

[C#] public object this[int columnIndex, DataRowVersion version] {get;}

[C++] public: \_\_property Object\* get\_Item(int columnIndex, DataRowVersion  
version);

[VB] Public Default ReadOnly Property Item(ByVal columnIndex As Integer,  
ByVal version As DataRowVersion) As Object

[JScript] returnValue = DataRowObject.Item(columnIndex, version);

### *Description*

Gets the data stored in the column, specified by index and version of the  
data to retrieve.

You can only create or update a row after calling the  
**System.Data.DataRow.BeginEdit** method; similarly, the  
**System.Data.DataRow.EndEdit** method must be called to commit the edit. After  
calling the **System.Data.DataRow.EndEdit** method, and before calling the

**System.Data.DataRow.AcceptChanges** method, internal representations of the original and new proposed values are stored. Therefore, until you call the **System.Data.DataRow.AcceptChanges** , you can use the *version* argument to specify which version of a column's value you need, either the **DataRowVersion.Original** or **DataRowVersion.Proposed** . Once you call the **System.Data.DataRow.AcceptChanges** method, however, the version of the column reverts to **DataRowVersion.Original** . If the row is new, you can also pass **DataRowVersion.Default** for the parameter to retrieve the column's default value. When passing **DataRowVersion.Current** , the property will return the current value, whatever its version may be. The zero-based index of the column. One of the **System.Data.DataRowVersion** values that specifies the desired row version. Possible values are **Default**, **Original**, **Current**, and **Proposed**.

ItemArray

ToString

[C#] public object[] ItemArray {get; set;}

[C++] public: \_\_property Object\* get\_ItemArray();public: \_\_property void

set\_ItemArray(Object\* \_\_gc[]);

[VB] Public Property ItemArray As Object ()

[JScript] public function get ItemArray() : Object[];public function set

ItemArray(Object[]);

### *Description*

Gets or sets all of the values for this row through an array.

If a **System.Data.DataColumn** has its **System.Data.DataColumn.DefaultValue** property set, pass a **null** in the array to set the default value for that column. Similarly, if a column has its **System.Data.DataColumn.AutoIncrement** property set to true, pass the **null** in the array to set the automatically generated value for the row.

RowError

ToString

[C#] public string RowError {get; set;}

[C++] public: \_\_property String\* get\_RowError();public: \_\_property void set\_RowError(String\*);

[VB] Public Property RowError As String

[JScript] public function get RowError() : String;public function set RowError(String);

### *Description*

Gets or sets the custom error description for a row.

Uses the **System.Data.DataRow.HasErrors** property to first determine if a **System.Data.DataRow** contains errors.

RowState

ToString

[C#] public DataRowState RowState {get;}

[C++] public: \_\_property DataRowState get\_RowState();

[VB] Public ReadOnly Property RowState As DataRowState

1 [JScript] public function get RowState() : DataRowState;

2

3 *Description*

4 Gets the current state of the row in regards to its relationship to the  
5 **System.Data.DataRowCollection** .

6 The **System.Data.DataRow.RowState** property is used in conjunction with  
7 the **System.Data.DataSet.GetChanges** and **System.Data.DataSet.HasChanges**  
8 methods of the **System.Data.DataSet** .

9 Table

10 ToString

11

12 [C#] public DataTable Table {get;}

13 [C++] public: \_\_property DataTable\* get\_Table();

14 [VB] Public ReadOnly Property Table As DataTable

15 [JScript] public function get Table() : DataTable;

16

17 *Description*

18 Gets the **System.Data.DataTable** for which this row has a schema.

19 A **System.Data.DataRow** does not necessarily belong to any table's  
20 collection of rows. This occurs when the **System.Data.DataRow** has been created  
21 but not added to the **System.Data.DataRowCollection** . If the  
22 **System.Data.DataRow.RowState** property returns **DataRowState.Detached** ,  
23 the row is not in any collection.

24 AcceptChanges

25

```

1
2 [C#] public void AcceptChanges();
3 [C++] public: void AcceptChanges();
4 [VB] Public Sub AcceptChanges()
5 [JScript] public function AcceptChanges();
6

```

### *Description*

Commits all the changes made to this row since the last time **System.Data.DataRow.AcceptChanges** was called.

When invoking **System.Data.DataRow.AcceptChanges** , the **System.Data.DataRow.EndEdit** method is implicitly called to end any edits. If the **System.Data.DataRow.RowState** of the row was **Added** or **Modified** , the **System.Data.DataRow.RowState** becomes **Unchanged** . If the **System.Data.DataRow.RowState** was **Deleted** , the row is removed.

### *BeginEdit*

```

17 [C#] public void BeginEdit();
18 [C++] public: void BeginEdit();
19 [VB] Public Sub BeginEdit()
20 [JScript] public function BeginEdit();
21

```

### *Description*

Begins an edit operation on a **System.Data.DataRow** object.

Use the **System.Data.DataRow.BeginEdit** method to put a **System.Data.DataRow** into edit mode. In this mode, events are temporarily

suspended allowing the user to make multiple changes to more than one row without triggering validation rules. For example, if the values of several rows must add up to 100, you can put each of the rows into edit mode to suspend the validation of the row values until the user attempts to commit the values. While in edit mode, the **The System.Data.DataRow.BeginEdit** method is called implicitly when the user changes the value of a databound control; the **System.Data.DataRow.EndEdit** method is called implicitly when you invoke the **System.Data.DataTable** object's **System.Data.DataTable.AcceptChanges** method.) While in this edit mode, the **System.Data.DataRow** stores representations of the original and new proposed values. Therefore, as long as the **System.Data.DataRow.EndEdit** method has not been called, you can retrieve either the original or proposed version by passing either **DataRowVersion.Original** or **DataRowVersion.Proposed** for the *version* parameter of the **System.Data.DataRow.Item(System.Int32)** property. You can also cancel any edits at this time by invoking the **System.Data.DataRow.CancelEdit** method.

CancelEdit

[C#] public void CancelEdit();

[C++] public: void CancelEdit();

[VB] Public Sub CancelEdit()

[JScript] public function CancelEdit();

#### *Description*

Cancels the current edit on the row.

See the **System.Data.DataRow.BeginEdit** method for more details.

**ClearErrors**

[C#] public void ClearErrors();

[C++] public: void ClearErrors();

[VB] Public Sub ClearErrors()

[JScript] public function ClearErrors();

*Description*

Clears the errors for the row, including the **System.Data.DataRow.RowError** and errors set with **System.Data.DataRow.SetColumnError(System.Int32,System.String)**.

*Use*

**System.Data.DataRow.SetColumnError(System.Int32,System.String)** and **System.Data.DataRow.GetColumnError(System.Int32)** to set and return errors for individual columns.

**Delete**

[C#] public void Delete();

[C++] public: void Delete();

[VB] Public Sub Delete()

[JScript] public function Delete();

*Description*

Deletes the row.

If the **System.Data.DataRow.RowState** of the row is **Added** , the row will be removed from the table.

EndEdit

[C#] public void EndEdit();

[C++] public: void EndEdit();

[VB] Public Sub EndEdit()

[JScript] public function EndEdit();

#### *Description*

Ends the edit occurring on the row.

When setting the property, an exception is generated if an exception occurs in the **System.Data.DataTable.RowChanging** event.

GetChildRows

[C#] public DataRow[] GetChildRows(DataRelation relation);

[C++] public: DataRow\* GetChildRows(DataRelation\* relation) [];

[VB] Public Function GetChildRows(ByVal relation As DataRelation) As

DataRow()

[JScript] public function GetChildRows(relation : DataRelation) : DataRow[];

Gets the child rows of a **System.Data.DataRow** .

#### *Description*

Gets the child rows of this **System.Data.DataRow** using the specified **System.Data.DataRelation** .

*Return Value:* An array of **System.Data.DataRow** objects (or an array of length zero).

In a **System.Data.DataSet** , the collection of all **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetChildRelations** method. The **System.Data.DataRelation** to use.

**GetChildRows**

[C#] public DataRow[] GetChildRows(string relationName);

[C++] public: DataRow\* GetChildRows(String\* relationName) [];

[VB] Public Function GetChildRows(ByVal relationName As String) As

DataRow()

[JScript] public function GetChildRows(relationName : String) : DataRow[]; Gets the child rows in a related **System.Data.DataTable** of a **System.Data.DataRow** .

### *Description*

Gets the child rows of a **System.Data.DataRow** using the specified **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation** .

*Return Value:* An array of **System.Data.DataRow** objects (or an array of length zero).

In a **System.Data.DataSet** , the collection of all **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetChildRelations** method. The **System.Data.DataRelation.RelationName** of the **System.Data.DataRelation** to use.

## GetChildRows

```
[C#] public DataRow[] GetChildRows(DataRelation relation, DataRowVersion
version);
[C++] public: DataRow* GetChildRows(DataRelation* relation, DataRowVersion
version) [];
[VB] Public Function GetChildRows(ByVal relation As DataRelation, ByVal
version As DataRowVersion) As DataRow()
[JScript] public function GetChildRows(relation : DataRelation, version :
DataRowVersion) : DataRow[];
```

### *Description*

Gets the child rows of a **System.Data.DataRow** using the specified **System.Data.DataRelation** , and **System.Data.DataRowVersion** .

*Return Value:* An array of **System.Data.DataRow** objects.

In a **System.Data.DataSet** , the collection of all **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetChildRelations** method. The **System.Data.DataRelation** to use. One of the **System.Data.DataRowVersion** values specifying the version of the data to get. Possible values are **Default**, **Original**, **Current**, and **Proposed** .

## GetChildRows

```
[C#] public DataRow[] GetChildRows(string relationName, DataRowVersion
version);
```

```

1  [C++] public: DataRow* GetChildRows(String* relationName, DataRowVersion
2  version) [];
3  [VB] Public Function GetChildRows(ByVal relationName As String, ByVal
4  version As DataRowVersion) As DataRow()
5  [JScript] public function GetChildRows(relationName : String, version :
6  DataRowVersion) : DataRow[];
7

```

### 8 *Description*

9 Gets the specified version of the child rows of a **System.Data.DataRow**  
 10 using the specified **System.Data.DataRelation.RelationName** of a  
 11 **System.Data.DataRelation** , and **System.Data.DataRowVersion** .

12 *Return Value:* An array of **System.Data.DataRow** objects (or an array of length  
 13 zero).

14 In a **System.Data.DataSet** , the collection of all  
 15 **System.Data.DataRelation** objects for the data set is returned by the  
 16 **System.Data.DataSet.GetChildRelations** method. The  
 17 **System.Data.DataRelation.RelationName** of the **System.Data.DataRelation** to  
 18 use. One of the **System.Data.DataRowVersion** values specifying the version of  
 19 the data to get. Possible values are **Default**, **Original**, **Current**, and **Proposed**.

### 20 *GetColumnError*

```

21
22 [C#] public string GetColumnError(DataColumn column);
23 [C++] public: String* GetColumnError(DataColumn* column);
24 [VB] Public Function GetColumnError(ByVal column As DataColumn) As String
25 [JScript] public function GetColumnError(column : DataColumn) : String;

```

## Description

Gets the error description of the specified **System.Data.DataColumn**.

*Return Value:* The text of the error description.

Use the

**System.Data.DataRow.SetColumnError(System.Int32,System.String)** method to set column errors. A **System.Data.DataColumn**.

**GetColumnError**

[C#] public string GetColumnError(int columnIndex);

[C++] public: String\* GetColumnError(int columnIndex);

[VB] Public Function GetColumnError(ByVal columnIndex As Integer) As String

[JScript] public function GetColumnError(columnIndex : int) : String; Gets the error description for a column.

## Description

Gets the error description for the column specified by index.

*Return Value:* The text of the error description.

Use the

**System.Data.DataRow.SetColumnError(System.Int32,System.String)** method to set column errors. The zero-based index of the column.

**GetColumnError**

[C#] public string GetColumnError(string columnName);

[C++] public: String\* GetColumnError(String\* columnName);

[VB] Public Function GetColumnError(ByVal columnName As String) As String  
 [JScript] public function GetColumnError(columnName : String) : String;

#### *Description*

Gets the error description for a column, specified by name.

*Return Value:* The text of the error description.

Use the

**System.Data.DataRow.SetColumnError(System.Int32,System.String)** method  
 to set column errors. The name of the column.

#### **GetColumnsInError**

[C#] public DataColumn[] GetColumnsInError();  
 [C++] public: DataColumn\* GetColumnsInError() [];  
 [VB] Public Function GetColumnsInError() As DataColumn()  
 [JScript] public function GetColumnsInError() : DataColumn[];

#### *Description*

Gets an array of columns that have errors.

*Return Value:* An array of **System.Data.DataColumn** objects that contain errors.

The **System.Data.DataRow.GetColumnsInError** allows you to reduce  
 the number of **System.Data.DataColumn** objects that must be processed for  
 errors by returning only those columns that have an error. Errors can be set to  
 individual columns with the  
**System.Data.DataRow.SetColumnError(System.Int32,System.String)** method.  
 To further reduce the number of processing, check the **System.Data.DataRow**

class's **System.Data.DataRow.HasErrors** property to first determine if a **System.Data.DataRow** has errors before invoking **System.Data.DataRow.GetColumnsInError** .

**GetParentRow**

[C#] public DataRow GetParentRow(DataRelation relation);  
 [C++] public: DataRow\* GetParentRow(DataRelation\* relation);  
 [VB] Public Function GetParentRow(ByVal relation As DataRelation) As DataRow  
 [JScript] public function GetParentRow(relation : DataRelation) : DataRow;

### *Description*

Gets the parent row of a **System.Data.DataRow** using the specified **System.Data.DataRelation** .

*Return Value:* The parent **System.Data.DataRow** of the current row.

In a **System.Data.DataSet** , the collection of all parent **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method. The **System.Data.DataRelation** to use.

**GetParentRow**

[C#] public DataRow GetParentRow(string relationName);  
 [C++] public: DataRow\* GetParentRow(String\* relationName);  
 [VB] Public Function GetParentRow(ByVal relationName As String) As DataRow  
 [JScript] public function GetParentRow(relationName : String) : DataRow; Gets

the parent row of a **System.Data.DataRow** .

### *Description*

Gets the parent row of a **System.Data.DataRow** using the specified **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation** .

*Return Value:* The parent **System.Data.DataRow** of the current row.

In a **System.Data.DataSet** , the collection of all parent **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method. The **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation**.

### **GetParentRow**

[C#] public DataRow GetParentRow(DataRelation relation, DataRowVersion version);

[C++] public: DataRow\* GetParentRow(DataRelation\* relation, DataRowVersion version);

[VB] Public Function GetParentRow(ByVal relation As DataRelation, ByVal version As DataRowVersion) As DataRow

[JScript] public function GetParentRow(relation : DataRelation, version : DataRowVersion) : DataRow;

### *Description*

Gets the parent row of a **System.Data.DataRow** using the specified **System.Data.DataRelation** , and **System.Data.DataRowVersion** .

*Return Value:* The parent **System.Data.DataRow** of the current row.

In a **System.Data.DataSet** , the collection of all parent **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method. The **System.Data.DataRelation** to use. One of the **System.Data.DataRowVersion** values specifying the version of the data to get.

#### GetParentRow

```
[C#] public DataRow GetParentRow(string relationName, DataRowVersion
version);
[C++] public: DataRow* GetParentRow(String* relationName, DataRowVersion
version);
[VB] Public Function GetParentRow(ByVal relationName As String, ByVal
version As DataRowVersion) As DataRow
[JScript] public function GetParentRow(relationName : String, version :
DataRowVersion) : DataRow;
```

#### *Description*

Gets the parent row of a **System.Data.DataRow** using the specified **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation** , and **System.Data.DataRowVersion** .

*Return Value:* The parent **System.Data.DataRow** of the current row.

In a **System.Data.DataSet** , the collection of all parent **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method.

The **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation**.

One of the **System.Data.DataRowVersion** values.

**GetParentRows**

[C#] public DataRow[] GetParentRows(DataRelation relation);

[C++] public: DataRow\* GetParentRows(DataRelation\* relation) [];

[VB] Public Function GetParentRows(ByVal relation As DataRelation) As  
DataRow()

[JScript] public function GetParentRows(relation : DataRelation) : DataRow[];

Gets the parent rows of a **System.Data.DataRow** .

### *Description*

Gets the parent rows of a **System.Data.DataRow** using the specified  
**System.Data.DataRelation** .

*Return Value:* An array of **System.Data.DataRow** objects (or an array of length  
zero).

In a **System.Data.DataSet** , the collection of all parent  
**System.Data.DataRelation** objects for the data set is returned by the  
**System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method.  
The **System.Data.DataRelation** to use.

**GetParentRows**

[C#] public DataRow[] GetParentRows(string relationName);

[C++] public: DataRow\* GetParentRows(String\* relationName) [];

[VB] Public Function GetParentRows(ByVal relationName As String) As

1 DataRow()

2 [JScript] public function GetParentRows(relationName : String) : DataRow[]; Gets  
3 the parent rows of a **System.Data.DataRow** .

4  
5 *Description*

6 Gets the parent rows of a **System.Data.DataRow** using the specified  
7 **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation** .

8 *Return Value:* An array of **System.Data.DataRow** objects (or an array of length  
9 zero).

10 In a **System.Data.DataSet** , the collection of all parent  
11 **System.Data.DataRelation** objects for the data set is returned by the  
12 **System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method.  
13 The **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation**.

14 **GetParentRows**

15  
16 [C#] public DataRow[] GetParentRows(DataRelation relation, DataRowVersion  
17 version);

18 [C++] public: DataRow\* GetParentRows(DataRelation\* relation,  
19 DataRowVersion version) [];

20 [VB] Public Function GetParentRows(ByVal relation As DataRelation, ByVal  
21 version As DataRowVersion) As DataRow()

22 [JScript] public function GetParentRows(relation : DataRelation, version :  
23 DataRowVersion) : DataRow[];

24  
25 *Description*

Gets the parent rows of a **System.Data.DataRow** using the specified **System.Data.DataRelation** , and **System.Data.DataRowVersion** .  
*Return Value:* An array of **System.Data.DataRow** objects (or an array of length zero).

In a **System.Data.DataSet** , the collection of all parent **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method. The **System.Data.DataRelation** to use. One of the **System.Data.DataRowVersion** values specifying the version of the data to get.

#### GetParentRows

```
[C#] public DataRow[] GetParentRows(string relationName, DataRowVersion
version);
[C++] public: DataRow* GetParentRows(String* relationName, DataRowVersion
version) [];
[VB] Public Function GetParentRows(ByVal relationName As String, ByVal
version As DataRowVersion) As DataRow()
[JScript] public function GetParentRows(relationName : String, version :
DataRowVersion) : DataRow[];
```

#### Description

Gets the parent rows of a **System.Data.DataRow** using the specified **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation** , and **System.Data.DataRowVersion** .

*Return Value:* An array of **System.Data.DataRow** objects (or an array of length zero).

In a **System.Data.DataSet**, the collection of all parent **System.Data.DataRelation** objects for the data set is returned by the **System.Data.DataSet.GetParentRelations(System.Data.DataTable)** method. The **System.Data.DataRelation.RelationName** of a **System.Data.DataRelation**. One of the **System.Data.DataRowVersion** values specifying the version of the data to get. Possible values are **Default**, **Original**, **Current**, and **Proposed**.

HasVersion

[C#] public bool HasVersion(DataRowVersion version);

[C++] public: bool HasVersion(DataRowVersion version);

[VB] Public Function HasVersion(ByVal version As DataRowVersion) As Boolean

[JScript] public function HasVersion(version : DataRowVersion) : Boolean;

*Description*

Gets a value indicating whether a specified version exists.

*Return Value:* **true** if the version exists; **otherwise**, false.

See the **System.Data.DataRow.BeginEdit** method for more details. One of the **System.Data.DataRowVersion** values that specifies the row version. Possible values are **Default**, **Original**, **Current**, and **Proposed**.

IsNull

[C#] public bool IsNull(DataColumn column);

```

1 [C++] public: bool IsNull(DataColumn* column);
2 [VB] Public Function IsNull(ByVal column As DataColumn) As Boolean
3 [JScript] public function IsNull(column : DataColumn) : Boolean;

```

#### 4 *Description*

5 Gets a value indicating whether the specified **System.Data.DataColumn**  
6 contains a null value.

7 *Return Value:* **true** if the column contains a null value; otherwise, **false** . A

8 **System.Data.DataColumn.**

9 **IsNull**

```

10
11
12 [C#] public bool IsNull(int columnIndex);
13 [C++] public: bool IsNull(int columnIndex);
14 [VB] Public Function IsNull(ByVal columnIndex As Integer) As Boolean
15 [JScript] public function IsNull(columnIndex : int) : Boolean; Gets a value
16 indicating whether the specified column contains a null value.

```

#### 17 *Description*

18 Gets a value indicating whether the column at the specified index contains a  
19 null value.

20 *Return Value:* **true** if the column contains a null value; otherwise, **false** . The zero-  
21 based index of the column.

22 **IsNull**

```

23
24
25 [C#] public bool IsNull(string columnName);

```

[C++] public: bool IsNull(String\* columnName);

[VB] Public Function IsNull(ByVal columnName As String) As Boolean

[JScript] public function IsNull(columnName : String) : Boolean;

### Description

Gets a value indicating whether the named column contains a null value.

**Return Value:** **true** if the column contains a null value; otherwise, **false** . The name of the column.

### IsNull

[C#] public bool IsNull(DataColumn column, DataRowVersion version);

[C++] public: bool IsNull(DataColumn\* column, DataRowVersion version);

[VB] Public Function IsNull(ByVal column As DataColumn, ByVal version As DataRowVersion) As Boolean

[JScript] public function IsNull(column : DataColumn, version : DataRowVersion) : Boolean;

### Description

Gets a value indicating whether the specified **System.Data.DataColumn** and **System.Data.DataRowVersion** contains a null value.

**Return Value:** **true** if the column contains a null value; otherwise, **false** . A **System.Data.DataColumn**. One of the **System.Data.DataRowVersion** values that specifies the row version. Possible values are **Default**, **Original**, **Current**, and **Proposed**.

### RejectChanges

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public void RejectChanges();
[C++] public: void RejectChanges();
[VB] Public Sub RejectChanges()
[JScript] public function RejectChanges();
```

*Description*

Rejects all changes made to the row since **System.Data.DataRow.AcceptChanges** was last called.

When calling the **System.Data.DataRow.RejectChanges** method, the **System.Data.DataRow.CancelEdit** method is implicitly called to cancel any edits. If **System.Data.DataRow.RowState** is **Deleted** or **Modified** , the row reverts to its previous values, and **System.Data.DataRow.RowState** becomes **Unchanged** . If the **System.Data.DataRow.RowState** is **Added** , the row is removed.

**SetColumnError**

```
[C#] public void SetColumnError(DataColumn column, string error);
[C++] public: void SetColumnError(DataColumn* column, String* error);
[VB] Public Sub SetColumnError(ByVal column As DataColumn, ByVal error As String)
[JScript] public function SetColumnError(column : DataColumn, error : String);
```

*Description*

Sets the error description for a column specified as a

**System.Data.DataColumn** .

To examine error descriptions, use the

**System.Data.DataRow.GetColumnError(System.Int32)** method. The

**System.Data.DataColumn** to set the error description for. The error description.

**SetColumnError**

[C#] public void SetColumnError(int columnIndex, string error);

[C++] public: void SetColumnError(int columnIndex, String\* error);

[VB] Public Sub SetColumnError(ByVal columnIndex As Integer, ByVal error As String)

[JScript] public function SetColumnError(columnIndex : int, error : String); Sets the error description for a column.

### *Description*

Sets the error description for a column specified by index.

The method is used to set custom error descriptions on specified columns.

You can use the **System.Windows.Forms.ErrorProvider** control to display the text of the error, or through by other reporting mechanisms. The zero-based index of the column. The error description.

**SetColumnError**

[C#] public void SetColumnError(string columnName, string error);

[C++] public: void SetColumnError(String\* columnName, String\* error);

[VB] Public Sub SetColumnError(ByVal columnName As String, ByVal error As

String)

[JScript] public function SetColumnError(columnName : String, error : String);

#### *Description*

Sets the error description for a column specified by name.

The name of a column is set with the **System.Data.DataColumn** class's **System.Data.DataColumn.ColumnName** property. The name of the column.  
The error description.

#### **SetNull**

[C#] protected void SetNull(DataColumn column);

[C++] protected: void SetNull(DataColumn\* column);

[VB] Protected Sub SetNull(ByVal column As DataColumn)

[JScript] protected function SetNull(column : DataColumn);

#### *Description*

Sets the the value of the specified **System.Data.DataColumn** to a null value.

Use the **System.Data.DataRow.IsNull(System.Int32)** method to determine if a column contains a null value. A **System.Data.DataColumn**.

#### **SetParentRow**

[C#] public void SetParentRow(DataRow parentRow);

[C++] public: void SetParentRow(DataRow\* parentRow);

[VB] Public Sub SetParentRow(ByVal parentRow As DataRow)

[JScript] public function SetParentRow(parentRow : DataRow); Sets the parent row of a **System.Data.DataRow** .

#### *Description*

Sets the parent row of a **System.Data.DataRow** with specified new parent **System.Data.DataRow** . The new parent **System.Data.DataRow** .

SetParentRow

[C#] public void SetParentRow(DataRow parentRow, DataRelation relation);

[C++] public: void SetParentRow(DataRow\* parentRow, DataRelation\* relation);

[VB] Public Sub SetParentRow(ByVal parentRow As DataRow, ByVal relation As DataRelation)

[JScript] public function SetParentRow(parentRow : DataRow, relation : DataRelation);

#### *Description*

Sets the parent row of a **System.Data.DataRow** with specified new parent **System.Data.DataRow** and **System.Data.DataRelation** .

[Need explanation of why we do this.] The following example sets the parent row of a given child row. The new parent **System.Data.DataRow** . The relation **System.Data.DataRelation** to use.

DataRowAction enumeration (System.Data)

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Describes the action taken on a **System.Data.DataRow** .  
Use the members of this enumeration to determine the action that has occurred on a **System.Data.DataRow** with respect to the **System.Data.DataTable** to which it belongs.

ToString

```
[C#] public const DataRowAction Add;
[C++] public: const DataRowAction Add;
[VB] Public Const Add As DataRowAction
[JScript] public var Add : DataRowAction;
```

*Description*

The row has been added to the table.

ToString

```
[C#] public const DataRowAction Change;
[C++] public: const DataRowAction Change;
[VB] Public Const Change As DataRowAction
[JScript] public var Change : DataRowAction;
```

*Description*

The row has changed.

1           ToString  
2  
3   [C#] public const DataRowAction Commit;  
4   [C++] public: const DataRowAction Commit;  
5   [VB] Public Const Commit As DataRowAction  
6   [JScript] public var Commit : DataRowAction;

7  
8   *Description*

9           The row has been committed.

10          ToString

11  
12   [C#] public const DataRowAction Delete;  
13   [C++] public: const DataRowAction Delete;  
14   [VB] Public Const Delete As DataRowAction  
15   [JScript] public var Delete : DataRowAction;

16  
17   *Description*

18           The row was deleted from the table.

19          ToString

20  
21   [C#] public const DataRowAction Nothing;  
22   [C++] public: const DataRowAction Nothing;  
23   [VB] Public Const Nothing As DataRowAction  
24   [JScript] public var Nothing : DataRowAction;

25

*Description*

The row has not changed.

ToString

[C#] public const DataRowAction Rollback;

[C++] public: const DataRowAction Rollback;

[VB] Public Const Rollback As DataRowAction

[JScript] public var Rollback : DataRowAction;

*Description*

The change has been rolled back.

DataRowBuilder class (System.Data)

ToString

*Description*

DataRowChangeEventArgs class (System.Data)

ToString

*Description*

Provides data for the **System.Data.DataTable.RowChanged** ,

**System.Data.DataTable.RowChanging** ,

**System.Data.DataTable.OnRowDeleting(System.Data.DataRowChangeEvent**

1 **Args)** , and

2 **System.Data.DataTable.OnRowDeleted(System.Data.DataRowChangeEvent**  
3 **Args)** events.

4 The **System.Data.DataTable.RowChanged** ,  
5 **System.Data.DataTable.RowChanged** , **System.Data.DataTable.RowChanged**  
6 , and **System.Data.DataTable.RowChanged** events occur when an action is  
7 performed on a **System.Data.DataRow** .

8 **DataRowChangeEventArgs**

9 *Example Syntax:*

10 **ToString**

11  
12 **[C#]** public **DataRowChangeEventArgs**(DataRow row, DataRowAction action);

13 **[C++]** public: **DataRowChangeEventArgs**(DataRow\* row, DataRowAction  
14 action);

15 **[VB]** Public Sub New(ByVal row As DataRow, ByVal action As DataRowAction)

16 **[JScript]** public function **DataRowChangeEventArgs**(row : DataRow, action :  
17 DataRowAction);

### 19 *Description*

20 Initializes a new instance of the **System.Data.DataRowChangeEventArgs**  
21 class. The **System.Data.DataRow** upon which an action is occurring. One of the  
22 **System.Data.DataRowAction** values.

23 **Action**

24 **ToString**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public DataRowAction Action {get;}
[C++] public: __property DataRowAction get_Action();
[VB] Public ReadOnly Property Action As DataRowAction
[JScript] public function get Action() : DataRowAction;
```

*Description*

Gets the action that has occurred on a **System.Data.DataRow** .  
Row  
ToString

```
[C#] public DataRow Row {get;}
[C++] public: __property DataRow* get_Row();
[VB] Public ReadOnly Property Row As DataRow
[JScript] public function get Row() : DataRow;
```

*Description*

Gets the row upon which an action has occurred.  
DataRowChangeEventHandler delegate (System.Data)  
ToString

*Description*

Represents the method that will handle the  
**System.Data.DataTable.RowChanging** ,

**System.Data.DataTable.RowChanged** , **System.Data.DataTable.RowDeleting**  
, and **System.Data.DataTable.RowDeleted** events of a **System.Data.DataTable** .

The source of the event. A **System.Data.DataRowChangeEventArgs** that  
contains the event data.

When you create a **System.Data.DataRowChangeEventHandler**  
delegate, you identify the method that will handle the event. To associate the event  
with your event handler, add an instance of the delegate to the event. The event  
handler is called whenever the event occurs, until you remove the delegate. For  
more information about delegates, see .

**DataRowCollection** class (**System.Data**)

**ToString**

### *Description*

Represents a collection of rows for a **System.Data.DataTable** .

The **System.Data.DataRowCollection** is a major component of the  
**System.Data.DataTable** . While the **System.Data.DataColumnCollection**  
defines the schema of the table, the **System.Data.DataRowCollection** contains  
the actual data for the table, where each **System.Data.DataRow** in the  
**System.Data.DataRowCollection** represents a single row.

**Count**

**IsReadOnly**

**IsSynchronized**

**Item**

**ToString**

*Description*

Gets the row at the specified index.

Use the **System.Data.DataRowCollection.Contains(System.Object)** method to determine if a given value exists in the key column of a row. The zero-based index of the row to return.

List

ToString

[C#] protected override ArrayList List {get;}

[C++] protected: \_\_property virtual ArrayList\* get\_List();

[VB] Overrides Protected ReadOnly Property List As ArrayList

[JScript] protected function get List() : ArrayList;

*Description*

Gets the list of the collection items.

SyncRoot

Add

[C#] public void Add(DataRow row);

[C++] public: void Add(DataRow\* row);

[VB] Public Sub Add(ByVal row As DataRow)

[JScript] public function Add(row : DataRow); Adds a **System.Data.DataRow** to the **System.Data.DataRowCollection** .

## *Description*

Adds the specified **System.Data.DataRow** to the **System.Data.DataRowCollection** object.

To create a new **System.Data.DataRow** , you must use the **System.Data.DataTable** class's **System.Data.DataTable.NewRow** method. When you use the **System.Data.DataTable.NewRow** method, a new **System.Data.DataRow** object is returned using the schema of parent **System.Data.DataTable** . After you create the **System.Data.DataRow** object and set the values for each of its columns, use the **System.Data.DataRowCollection.Add(System.Data.DataRow)** method to add the object to the collection. The **System.Data.DataRow** to add.

### Add

```
[C#] public virtual DataRow Add(object[] values);
[C++] public: virtual DataRow* Add(Object* values __gc[]);
[VB] Overridable Public Function Add(ByVal values() As Object) As DataRow
[JScript] public function Add(values : Object[]) : DataRow;
```

## *Description*

Creates a row using specified values and adds it to the **System.Data.DataRowCollection** .

If a **System.Data.DataColumn** object has its **System.Data.DataColumn.AutoIncrement** set to **True**, **System.Object.Empty**

should be passed to get the default value for that column. The array of values that are used to create the new row.

### Clear

[C#] public void Clear();

[C++] public: void Clear();

[VB] Public Sub Clear()

[JScript] public function Clear();

### *Description*

Clears the collection of all rows.

To add a row to the collection, first use the **System.Data.DataTable** class's **System.Data.DataTable.NewRow** method to create the new row. Then add the new row using the **System.Data.DataRowCollection.Add(System.Data.DataRow)** method of the **System.Data.DataRowCollection** class.

### Contains

[C#] public bool Contains(object key);

[C++] public: bool Contains(Object\* key);

[VB] Public Function Contains(ByVal key As Object) As Boolean

[JScript] public function Contains(key : Object) : Boolean; Gets a value indicating whether any row in the collection contains a specified value in the primary key or keys column.

## Description

Gets a value indicating whether the primary key of any row in the collection contains the specified value.

**Return Value:** **true** if the collection contains a **System.Data.DataRow** with the specified primary key value; otherwise, **false** .

To use the **System.Data.DataRowCollection.Contains(System.Object)** method, the **System.Data.DataTable** object to which the **System.Data.DataRowCollection** object belongs to must have at least one column designated as a primary key column. See the **System.Data.DataTable.PrimaryKey** property for details on creating a primary key column. The value of the primary key to test for.

## Contains

[C#] public bool Contains(object[] keys);

[C++] public: bool Contains(Object\* keys \_\_gc[]);

[VB] Public Function Contains(ByVal keys() As Object) As Boolean

[JScript] public function Contains(keys : Object[]) : Boolean;

## Description

Gets a value indicating if the **System.Data.DataRow** with the specified primary key values exists.

**Return Value:** **true** if the **System.Data.DataRowCollection** contains a **System.Data.DataRow** with the specified key values; otherwise, **false** .

To use the **System.Data.DataRowCollection.Contains(System.Object)** method with an array of values, the **System.Data.DataTable** object to which the **System.Data.DataRowCollection** object belongs must have at an array of columns designated as a primary keys. See the **System.Data.DataTable.PrimaryKey** property for details on creating an array of primary key columns. The number of array elements must correspond to the number of primary key columns in the **System.Data.DataTable** . An array of primary key values to test for.

#### Find

[C#] public DataRow Find(object key);  
 [C++] public: DataRow\* Find(Object\* key);  
 [VB] Public Function Find(ByVal key As Object) As DataRow  
 [JScript] public function Find(key : Object) : DataRow; Gets a specified **System.Data.DataRow** .

#### Description

Gets the row specified by the primary key value.

**Return Value:** A **System.Data.DataRow** containing the primary key value specified; otherwise a null value if the primary key value does not exist in the **System.Data.DataRowCollection** .

To use the **System.Data.DataRowCollection.Contains(System.Object)** method, the **System.Data.DataTable** object to which the **System.Data.DataRowCollection** object belongs to must have at least one column designated as a primary key column. See the

**System.Data.DataTable.PrimaryKey** property for details on creating a primary key column. The primary key value of the **System.Data.DataRow** to find.

### Find

```
[C#] public DataRow Find(object[] keys);
[C++] public: DataRow* Find(Object* keys __gc[]);
[VB] Public Function Find(ByVal keys() As Object) As DataRow
[JScript] public function Find(keys : Object[]) : DataRow;
```

### Description

Gets the row containing the specified primary key values.

**Return Value:** An array of **System.Data.DataRow** objects containing the primary key values specified; otherwise a null value if the primary key values do not exist in the **System.Data.DataRowCollection**.

To use the **System.Data.DataRowCollection.Find(System.Object)** method, the **System.Data.DataTable** object to which the **System.Data.DataRowCollection** object belongs to must have at least one column designated as a primary key column. See the **System.Data.DataTable.PrimaryKey** property for details on creating a **System.Data.DataTable.PrimaryKey** column, or an array of **System.Data.DataColumn** objects when the table has more than one primary key. An array of primary key values to find. The type of the array is **Object**.

### InsertAt

```
[C#] public void InsertAt(DataRow row, int pos);
```

1 [C++] public: void InsertAt(DataRow\* row, int pos);

2 [VB] Public Sub InsertAt(ByVal row As DataRow, ByVal pos As Integer)

3 [JScript] public function InsertAt(row : DataRow, pos : int);

4  
5 *Description*

6  
7 Remove

8  
9 [C#] public void Remove(DataRow row);

10 [C++] public: void Remove(DataRow\* row);

11 [VB] Public Sub Remove(ByVal row As DataRow)

12 [JScript] public function Remove(row : DataRow); Removes a specific row from  
13 the **System.Data.DataRowCollection** .

14  
15 *Description*

16 Removes the specified **System.Data.DataRow** from the collection.

17 When a row is removed, data in that row is lost. You can also call the  
18 **System.Data.DataRow** class's **System.Data.DataRow.Delete** method to simply  
19 mark a row for removal. The row is not actually removed until the  
20 **System.Data.DataRow.AcceptChanges** method is invoked. The  
21 **System.Data.DataRow** to remove.

22 RemoveAt

23  
24 [C#] public void RemoveAt(int index);

25 [C++] public: void RemoveAt(int index);

[VB] Public Sub RemoveAt(ByVal index As Integer)

[JScript] public function RemoveAt(index : int);

#### *Description*

Removes the row with the specified index from the collection.

When a row is removed, data in that row is lost. You can also call the **System.Data.DataRow** class's **System.Data.DataRow.Delete** method to simply mark a row for removal. The row is not actually removed until the **System.Data.DataRow.AcceptChanges** method is invoked. The index of the row to remove.

DataRowState enumeration (System.Data)

ToString

#### *Description*

Gets the state of a **System.Data.DataRow** object.

The **System.Data.DataRowState** enumeration is returned by the **System.Data.DataRow.RowState** property of the **System.Data.DataRow** class.

ToString

[C#] public const DataRowState Added;

[C++] public: const DataRowState Added;

[VB] Public Const Added As DataRowState

[JScript] public var Added : DataRowState;

*Description*

The row has been added to a **System.Data.DataRowCollection** , and **System.Data.DataRow.AcceptChanges** has not been called.

ToString

[C#] public const DataRowState Deleted;  
[C++] public: const DataRowState Deleted;  
[VB] Public Const Deleted As DataRowState  
[JScript] public var Deleted : DataRowState;

*Description*

The row was deleted using the **System.Data.DataRow.Delete** method of the **System.Data.DataRow** .

ToString

[C#] public const DataRowState Detached;  
[C++] public: const DataRowState Detached;  
[VB] Public Const Detached As DataRowState  
[JScript] public var Detached : DataRowState;

*Description*

The row has been created but is not part of any **System.Data.DataRowCollection** . A **System.Data.DataRow** is in this state

1 immediately after it has been created and before it is added to a collection, or if it  
2 has been removed from a collection.

3 ToString

4  
5 [C#] public const DataRowState Modified;  
6 [C++] public: const DataRowState Modified;  
7 [VB] Public Const Modified As DataRowState  
8 [JScript] public var Modified : DataRowState;

9  
10 *Description*

11 The row has been modified and **System.Data.DataRow.AcceptChanges**  
12 has not been called.

13 ToString

14  
15 [C#] public const DataRowState Unchanged;  
16 [C++] public: const DataRowState Unchanged;  
17 [VB] Public Const Unchanged As DataRowState  
18 [JScript] public var Unchanged : DataRowState;

19  
20 *Description*

21 The row has not changed since **System.Data.DataRow.AcceptChanges**  
22 was last called.

23 DataRowVersion enumeration (System.Data)

24 ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Describes the version of a **System.Data.DataRow** .

The **System.Data.DataRowVersion** values are used when retrieving the value found in a **System.Data.DataRow** using **System.Data.DataRow.Item(System.Int32)** or the **System.Data.DataRow.GetChildRows(System.String)** of the **System.Data.DataRow** object.

**ToString**

[C#] public const DataRowVersion Current;  
[C++] public: const DataRowVersion Current;  
[VB] Public Const Current As DataRowVersion  
[JScript] public var Current : DataRowVersion;

*Description*

The row contains current values.

**ToString**

[C#] public const DataRowVersion Default;  
[C++] public: const DataRowVersion Default;  
[VB] Public Const Default As DataRowVersion  
[JScript] public var Default : DataRowVersion;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

The row contains its default values.

ToString

[C#] public const DataRowVersion Original;

[C++] public: const DataRowVersion Original;

[VB] Public Const Original As DataRowVersion

[JScript] public var Original : DataRowVersion;

*Description*

The row contains its original values.

ToString

[C#] public const DataRowVersion Proposed;

[C++] public: const DataRowVersion Proposed;

[VB] Public Const Proposed As DataRowVersion

[JScript] public var Proposed : DataRowVersion;

*Description*

The row contains a proposed value.

DataRowView class (System.Data)

ToString

### *Description*

Represents a customized view of a **System.Data.DataRow** exposed as a fully featured Windows Forms control.

Whenever data is displayed (for example in a **System.Windows.Forms.DataGrid** control), only one version of each row can be displayed. The displayed row is a **System.Data.DataRowView**.

**DataRowView**

**ToString**

[C#] public DataRowView DataRowView {get;}

[C++] public: \_\_property DataRowView\* get\_DataView();

[VB] Public ReadOnly Property DataRowView As DataRowView

[JScript] public function get DataRowView() : DataRowView;

### *Description*

Gets the **System.Data.DataView** to which this row belongs.

**IsEdit**

**ToString**

[C#] public bool IsEdit {get;}

[C++] public: \_\_property bool get\_IsEdit();

[VB] Public ReadOnly Property IsEdit As Boolean

[JScript] public function get IsEdit() : Boolean;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Indicates whether the row is in edit mode.

IsNew

ToString

```
[C#] public bool IsNew {get;}
[C++] public: __property bool get_IsNew();
[VB] Public ReadOnly Property IsNew As Boolean
[JScript] public function get IsNew() : Boolean;
```

*Description*

Indicates whether a **System.Data.DataRowView** is new.

Item

ToString

```
[C#] public object this[string property] {get; set;}
[C++] public: __property Object* get_Item(String* property);public: __property
void set_Item(String* property, Object*);
[VB] Public Default Property Item(ByVal property As String) As Object
[JScript] returnValue =
DataRowViewObject.Item(property);DataRowViewObject.Item(property) =
returnValue;
```

*Description*

Gets or sets a value in a specified column. String that contains the specified column.

Item

ToString

[C#] public object this[int ndx] {get; set;}

[C++] public: \_\_property Object\* get\_Item(int ndx);public: \_\_property void set\_Item(int ndx, Object\*);

[VB] Public Default Property Item(ByVal ndx As Integer) As Object

[JScript] returnValue =

DataRowViewObject.Item(ndx);DataRowViewObject.Item(ndx) = returnValue;

Gets or sets a value in a specified column.

#### *Description*

Gets or sets a value in a specified column. The specified column.

Row

ToString

[C#] public DataRow Row {get;}

[C++] public: \_\_property DataRow\* get\_Row();

[VB] Public ReadOnly Property Row As DataRow

[JScript] public function get Row() : DataRow;

#### *Description*

Gets the **System.Data.DataRow** being viewed.

RowVersion

ToString

[C#] public DataRowVersion RowVersion {get;}

[C++] public: \_\_property DataRowVersion get\_RowVersion();

[VB] Public ReadOnly Property RowVersion As DataRowVersion

[JScript] public function get RowVersion() : DataRowVersion;

### *Description*

Gets the current version description of the **System.Data.DataRow** .

For more details, see **System.Data.DataRowVersion** .

BeginEdit

[C#] public void BeginEdit();

[C++] public: \_\_sealed void BeginEdit();

[VB] NotOverridable Public Sub BeginEdit()

[JScript] public function BeginEdit();

### *Description*

Begins an edit procedure.

The **System.Data.DataRowView.BeginEdit** method is identical to the **System.Data.DataRow.BeginEdit** method of the **System.Data.DataRow** . After calling **System.Data.DataRowView.BeginEdit** , any changes made to the **System.Data.DataRowView** can be rolled back by calling **System.Data.DataRow.CancelEdit** . Call the

**System.Data.DataRowView.BeginEdit** method before allowing users to change row values. After values have been changed, you retrieve the new values by setting the **System.Data.DataRowView.RowVersion** to **DataRowVersion.Proposed** . Check the values with a business rule, and roll back the changes if needed by calling **System.Data.DataRowView.CancelEdit** , or call **System.Data.DataRowView.EndEdit** to accept the changes.

#### CancelEdit

```
[C#] public void CancelEdit();
[C++] public: __sealed void CancelEdit();
[VB] NotOverridable Public Sub CancelEdit()
[JavaScript] public function CancelEdit();
```

#### *Description*

Cancels an edit procedure.

After calling **System.Data.DataRowView.CancelEdit** , all changes to the row are rolled back. You can also roll back changes by calling **System.Data.DataTable.RejectChanges** on the parent **System.Data.DataTable** .

#### CreateChildView

```
[C#] public DataView CreateChildView(DataRelation relation);
[C++] public: DataView* CreateChildView(DataRelation* relation);
[VB] Public Function CreateChildView(ByVal relation As DataRelation) As
DataView
[JavaScript] public function CreateChildView(relation : DataRelation) : DataView;
```

1 Returns a **System.Data.DataView** for the child **System.Data.DataTable** .

2

3 *Description*

4 Returns a **System.Data.DataView** for the child **System.Data.DataTable**  
5 with the specified **DataRelation**. The **System.Data.DataRelation** object.

6 CreateChildView

7

8 [C#] public DataView CreateChildView(string relationName);

9 [C++] public: DataView\* CreateChildView(String\* relationName);

10 [VB] Public Function CreateChildView(ByVal relationName As String) As

11 DataView

12 [JScript] public function CreateChildView(relationName : String) : DataView;

13

14 *Description*

15 Returns a **System.Data.DataView** for the child **System.Data.DataTable**  
16 with the specified **DataRelation** name. A string containing the  
17 **System.Data.DataRelation** name.

18 Delete

19

20 [C#] public void Delete();

21 [C++] public: void Delete();

22 [VB] Public Sub Delete()

23 [JScript] public function Delete();

24

25 *Description*

1 Deletes a row.  
 2 The row is not permanently deleted until  
 3 **System.Data.DataTable.AcceptChanges** is invoked on the  
 4 **System.Data.DataTable** that row belongs to.

5 EndEdit

6  
 7 [C#] public void EndEdit();  
 8 [C++] public: \_\_sealed void EndEdit();  
 9 [VB] NotOverridable Public Sub EndEdit()  
 10 [JScript] public function EndEdit();  
 11

12 *Description*

13 Ends an edit procedure.

14 Equals

15  
 16 [C#] public override bool Equals(object other);  
 17 [C++] public: bool Equals(Object\* other);  
 18 [VB] Overrides Public Function Equals(ByVal other As Object) As Boolean  
 19 [JScript] public override function Equals(other : Object) : Boolean;  
 20

21 *Description*

22 Gets a value indicating whether the current **System.Data.DataRowView** is  
 23 identical to the specified object.

24 *Return Value:* **true** , if *object* is a **System.Data.DataRowView** and it returns the  
 25

1 same row as the current **System.Data.DataRowView** ; otherwise, **false** . An  
2 **System.Object** to be compared.

### 3 GetHashCode

4  
5 [C#] public override int GetHashCode();  
6 [C++] public: int GetHashCode();  
7 [VB] Overrides Public Function GetHashCode() As Integer  
8 [JScript] public override function GetHashCode() : int;

### 9 *Description*

10 Returns the hash code of the **System.Data.DataRow** object.

11  
12 *Return Value:* A 32-bit signed integer hash code, one, which represents Boolean  
13 **true** if the value of this instance is nonzero; otherwise, the integer, zero, which  
14 represents Boolean **false** .

### 15 ICustomTypeDescriptor.GetAttributes

16  
17 [C#] AttributeCollection ICustomTypeDescriptor.GetAttributes();  
18 [C++] AttributeCollection\* ICustomTypeDescriptor::GetAttributes();  
19 [VB] Function GetAttributes() As AttributeCollection Implements  
20 ICustomTypeDescriptor.GetAttributes  
21 [JScript] function ICustomTypeDescriptor.GetAttributes() : AttributeCollection;

### 22 ICustomTypeDescriptor.GetClassName

23  
24 [C#] string ICustomTypeDescriptor.GetClassName();  
25 [C++] String\* ICustomTypeDescriptor::GetClassName();

```

1  [VB] Function GetClassName() As String Implements
2  ICustomTypeDescriptor.GetClassName
3  [JScript] function ICustomTypeDescriptor.GetClassName() : String;
4      ICustomTypeDescriptor.GetComponentName
5
6  [C#] string ICustomTypeDescriptor.GetComponentName();
7  [C++] String* ICustomTypeDescriptor::GetComponentName();
8  [VB] Function GetComponentName() As String Implements
9  ICustomTypeDescriptor.GetComponentName
10 [JScript] function ICustomTypeDescriptor.GetComponentName() : String;
11     ICustomTypeDescriptor.GetConverter
12
13 [C#] TypeConverter ICustomTypeDescriptor.GetConverter();
14 [C++] TypeConverter* ICustomTypeDescriptor::GetConverter();
15 [VB] Function GetConverter() As TypeConverter Implements
16 ICustomTypeDescriptor.GetConverter
17 [JScript] function ICustomTypeDescriptor.GetConverter() : TypeConverter;
18     ICustomTypeDescriptor.GetDefaultEvent
19
20 [C#] EventDescriptor ICustomTypeDescriptor.GetDefaultEvent();
21 [C++] EventDescriptor* ICustomTypeDescriptor::GetDefaultEvent();
22 [VB] Function GetDefaultEvent() As EventDescriptor Implements
23 ICustomTypeDescriptor.GetDefaultEvent
24 [JScript] function ICustomTypeDescriptor.GetDefaultEvent() : EventDescriptor;
25     ICustomTypeDescriptor.GetDefaultProperty

```

```

1
2 [C#]PropertyDescriptor ICustomPropertyDescriptor.DefaultProperty();
3 [C++]PropertyDescriptor* ICustomPropertyDescriptor::GetDefaultProperty();
4 [VB]Function GetDefaultProperty() As PropertyDescriptor Implements
5 ICustomPropertyDescriptor.DefaultProperty
6 [JScript]function ICustomPropertyDescriptor.DefaultProperty() :
7 PropertyDescriptor;
8     ICustomPropertyDescriptor.GetEditor
9
10 [C#] object ICustomPropertyDescriptor.GetEditor(Type editorBaseType);
11 [C++] Object* ICustomPropertyDescriptor::GetEditor(Type* editorBaseType);
12 [VB]Function GetEditor(ByVal editorBaseType As Type) As Object Implements
13 ICustomPropertyDescriptor.GetEditor
14 [JScript]function ICustomPropertyDescriptor.GetEditor(editorBaseType : Type) :
15 Object;
16     ICustomPropertyDescriptor.GetEvents
17
18 [C#] EventDescriptorCollection ICustomPropertyDescriptor.GetEvents();
19 [C++] EventDescriptorCollection* ICustomPropertyDescriptor::GetEvents();
20 [VB]Function GetEvents() As EventDescriptorCollection Implements
21 ICustomPropertyDescriptor.GetEvents
22 [JScript]function ICustomPropertyDescriptor.GetEvents() :
23 EventDescriptorCollection;
24     ICustomPropertyDescriptor.GetEvents
25

```

```
1
2 [C#] EventDescriptorCollection ICustomTypeDescriptor.GetEvents(Attribute[]
3 attributes);
4 [C++] EventDescriptorCollection* ICustomTypeDescriptor::GetEvents(Attribute*
5 attributes[]);
6 [VB] Function GetEvents(ByVal attributes() As Attribute) As
7 EventDescriptorCollection Implements ICustomTypeDescriptor.GetEvents
8 [JScript] function ICustomTypeDescriptor.GetEvents(attributes : Attribute[]) :
9 EventDescriptorCollection;
10     ICustomTypeDescriptor.GetProperties
11
12 [C#] PropertyDescriptorCollection ICustomTypeDescriptor.GetProperties();
13 [C++] PropertyDescriptorCollection* ICustomTypeDescriptor::GetProperties();
14 [VB] Function GetProperties() As PropertyDescriptorCollection Implements
15 ICustomTypeDescriptor.GetProperties
16 [JScript] function ICustomTypeDescriptor.GetProperties() :
17 PropertyDescriptorCollection;
18     ICustomTypeDescriptor.GetProperties
19
20 [C#] PropertyDescriptorCollection
21 ICustomTypeDescriptor.GetProperties(Attribute[] attributes);
22 [C++] PropertyDescriptorCollection*
23 ICustomTypeDescriptor::GetProperties(Attribute* attributes[]);
24 [VB] Function GetProperties(ByVal attributes() As Attribute) As
25 PropertyDescriptorCollection Implements ICustomTypeDescriptor.GetProperties
```

```

1 [JScript] function ICustomPropertyDescriptor.GetProperties(attributes : Attribute[]) :
2   PropertyDescriptorCollection;
3     ICustomPropertyDescriptor.GetPropertyOwner
4
5 [C#] object ICustomPropertyDescriptor.GetPropertyOwner(PropertyDescriptor pd);
6 [C++] Object* ICustomPropertyDescriptor::GetPropertyOwner(PropertyDescriptor*
7   pd);
8 [VB] Function GetPropertyOwner(ByVal pd As PropertyDescriptor) As Object
9   Implements ICustomPropertyDescriptor.GetPropertyOwner
10 [JScript] function ICustomPropertyDescriptor.GetPropertyOwner(pd :
11   PropertyDescriptor) : Object;

```

```

12     DataSet class (System.Data)
13     ToString

```

## 16 *Description*

17 Represents an in-memory cache of data.

18 The **System.Data.DataSet** , which is an in-memory cache of data retrieved  
 19 from a database, is a major component of the ADO.NET architecture. The  
 20 **System.Data.DataSet** consists of a collection of **System.Data.DataTable** objects  
 21 that you can relate to each other with **System.Data.DataRelation** objects. You  
 22 can also enforce data integrity in the **System.Data.DataSet** by using the  
 23 **System.Data.UniqueConstraint** and **System.Data.ForeignKeyConstraint**  
 24 objects. For further details about working with **System.Data.DataSet** objects, see  
 25 .

DataSet

*Example Syntax:*

ToString

[C#] public DataSet();

[C++] public: DataSet();

[VB] Public Sub New()

[JScript] public function DataSet(); Initializes a new instance of the

**System.Data.DataSet** class.

*Description*

Initializes a new instance of the **System.Data.DataSet** class.

This implementation of the **System.Data.DataSet** constructor takes no parameters, and creates a default name, "NewDataSet", for the new instance.

DataSet

*Example Syntax:*

ToString

[C#] public DataSet(string dataSetName);

[C++] public: DataSet(String\* dataSetName);

[VB] Public Sub New(ByVal dataSetName As String)

[JScript] public function DataSet(dataSetName : String);

*Description*

1        Initializes a new instance of a **System.Data.DataSet** class with the given  
2        name.

3        A name for the **System.Data.DataSet** is required to ensure that the XML  
4        representation of the **System.Data.DataSet** always has a name for the document  
5        element, which is the highest level element in a schema definition. The name of  
6        the **System.Data.DataSet** .

7        DataSet

8        *Example Syntax:*

9        ToString

11      [C#] protected DataSet(SerializationInfo info, StreamingContext context);

12      [C++] protected: DataSet(SerializationInfo\* info, StreamingContext context);

13      [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As  
14      StreamingContext)

15      [JScript] protected function DataSet(info : SerializationInfo, context :  
16      StreamingContext);

### 18      *Description*

19      Initializes a new instance of the **System.Data.DataSet** class with the  
20      **System.Runtime.Serialization.SerializationInfo** and the  
21      **System.Runtime.Serialization.StreamingContext** .

22      This implemenation of the **System.Data.DataSet** constructor is required  
23      for **System.Runtime.Serialization.ISerializable** . The data needed to serialize or  
24      deserialize an object. The source and destination of a given serialized stream.

25      CaseSensitive

ToString

[C#] public bool CaseSensitive {get; set;}

[C++] public: \_\_property bool get\_CaseSensitive();public: \_\_property void  
set\_CaseSensitive(bool);

[VB] Public Property CaseSensitive As Boolean

[JScript] public function get CaseSensitive() : Boolean;public function set  
CaseSensitive(Boolean);

#### *Description*

Gets or sets a value indicating whether string comparisons within  
**System.Data.DataTable** objects are case-sensitive.

The **System.Data.DataSet.CaseSensitive** property affects how sorting,  
searching, and filtering operations are performed on each  
**System.Data.DataTable** contained in a **System.Data.DataSet** when using the  
**System.Data.DataTable.Select** method .

Container

DataSetName

ToString

#### *Description*

Gets or sets the name of the current **System.Data.DataSet** .

DefaultViewManager

ToString

```

1
2 [C#] public DataViewManager DefaultViewManager {get;}
3 [C++] public: __property DataViewManager* get_DefaultViewManager();
4 [VB] Public ReadOnly Property DefaultViewManager As DataViewManager
5 [JScript] public function get DefaultViewManager() : DataViewManager;
6

```

### *Description*

Gets a custom view of the data contained by the **System.Data.DataSet** that allows filtering, searching, and navigating using a custom **System.Data.DataViewManager** .

The **System.Data.DataViewManager** returned by the **System.Data.DataSet.DefaultViewManager** property allows you to create custom settings for each **System.Data.DataTable** in the **System.Data.DataSet** . When you add **System.Data.DataTable** objects to the **System.Data.DataTableCollection** , each table is automatically configured to display rows according to the specified property settings of the **System.Data.DataView** , including sort order, filtering, and **System.Data.DataViewRowState** .

DesignMode

EnforceConstraints

ToString

### *Description*

Gets or sets a value indicating whether constraint rules are followed when attempting any update operation.

See the **System.Data.DataTable.Constraints** property for more details.

Events

ExtendedProperties

ToString

### *Description*

Gets the collection of custom user information.

The **System.Data.DataSet.ExtendedProperties** property allows you to store custom information with the object. For example, you may store a time when the data should be refreshed.

HasErrors

ToString

[C#] public bool HasErrors {get;}

[C++] public: \_\_property bool get\_HasErrors();

[VB] Public ReadOnly Property HasErrors As Boolean

[JScript] public function get HasErrors() : Boolean;

### *Description*

Gets a value indicating whether there are errors in any of the rows in any of the tables of this **System.Data.DataSet**.

The **System.Data.DataSet.HasErrors** property is usually consulted after creating using the **System.Data.DataSet.GetChanges** method. Check the **System.Data.DataSet.HasErrors** property of the **System.Data.DataSet** created with the **System.Data.DataSet.GetChanges** method to determine if any errors exists. If so, you should reconcile the errors before proceeding to update the data source with the changes.

Locale

ToString

[C#] public CultureInfo Locale {get; set;}

[C++] public: \_\_property CultureInfo\* get\_Locale();public: \_\_property void set\_Locale(CultureInfo\*);

[VB] Public Property Locale As CultureInfo

[JScript] public function get Locale() : CultureInfo;public function set Locale(CultureInfo);

### *Description*

Gets or sets the locale information used to compare strings within the table.

The **System.Data.DataSet.Locale** property specifies the locale for which sorting will apply.

Namespace

ToString

[C#] public string Namespace {get; set;}

[C++] public: \_\_property String\* get\_Namespace();public: \_\_property void

1 set\_Namespace(String\*);

2 [VB] Public Property Namespace As String

3 [JScript] public function get Namespace() : String;public function set

4 Namespace(String);

6 *Description*

7 Gets or sets the namespace of the **System.Data.DataSet** .

8 The **System.Data.DataSet.Namespace** property is used when reading and  
9 writing an XML document into the **System.Data.DataSet** using the

10 **System.Data.DataSet.ReadXml(System.Xml.XmlReader)** ,

11 **System.Data.DataSet.WriteXml(System.IO.Stream)** ,

12 **System.Data.DataSet.ReadXmlSchema(System.Xml.XmlReader)** , or

13 **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** methods.

14 Prefix

15 ToString

17 [C#] public string Prefix {get; set;}

18 [C++] public: \_\_property String\* get\_Prefix();public: \_\_property void

19 set\_Prefix(String\*);

20 [VB] Public Property Prefix As String

21 [JScript] public function get Prefix() : String;public function set Prefix(String);

23 *Description*

24 Gets or sets an XML prefix that aliases the namespace of the

25 **System.Data.DataSet** .

The **System.Data.DataSet.Prefix** is used throughout an XML document to identify elements which belong to the **System.Data.DataSet** object's namespace (as set by the **System.Data.DataSet.Namespace** property).

Relations

ToString

[C#] public DataRelationCollection Relations {get;}

[C++] public: \_\_property DataRelationCollection\* get\_Relations();

[VB] Public ReadOnly Property Relations As DataRelationCollection

[JScript] public function get Relations() : DataRelationCollection;

### *Description*

Get the collection of relations that link tables and allow navigation from parent tables to child tables.

Site

ToString

[C#] public override ISite Site {get; set;}

[C++] public: \_\_property virtual ISite\* get\_Site();public: \_\_property virtual void set\_Site(ISite\*);

[VB] Overrides Public Property Site As ISite

[JScript] public function get Site() : ISite;public function set Site(ISite);

### *Description*

1 Gets or sets an **System.ComponentModel.ISite** for the  
2 **System.Data.DataSet** .

3 Sites bind a **System.ComponentModel.Component** to a  
4 **System.ComponentModel.Container** and enable communication between them,  
5 as well as provide a way for the container to manage its components.

6 Tables

7 ToString

8  
9 [C#] public DataTableCollection Tables {get;}

10 [C++] public: \_\_property DataTableCollection\* get\_ Tables();

11 [VB] Public ReadOnly Property Tables As DataTableCollection

12 [JScript] public function get Tables() : DataTableCollection;

13  
14 *Description*

15 Gets the collection of tables contained in the **System.Data.DataSet** .

16 To add tables to the collection, use

17 **System.Data.DataTableCollection.Add(System.Data.DataTable)** method of the

18 **System.Data.DataTableCollection** . To remove tables, use the

19 **System.Data.DataTableCollection.Remove(System.Data.DataTable)** method.

20 ToString

21  
22  
23 *Description*

24 Occurs when a target and source **System.Data.DataRow** have the same  
25 primary key value, and **System.Data.DataSet.EnforceConstraints** is set to true.

For more information about handling events, see .

AcceptChanges

[C#] public void AcceptChanges();

[C++] public: void AcceptChanges();

[VB] Public Sub AcceptChanges()

[JScript] public function AcceptChanges();

### *Description*

Commits all the changes made to this **System.Data.DataSet** since it was loaded or the last time **System.Data.DataSet.AcceptChanges** was called.

Both the **System.Data.DataRow** and **System.Data.DataTable** classes also have **System.Data.DataSet.AcceptChanges** methods. Invoking **System.Data.DataSet.AcceptChanges** on the **System.Data.DataSet** causes **System.Data.DataTable.AcceptChanges** to be called on each table in a **System.Data.DataSet** . Consequently, calling **System.Data.DataTable.AcceptChanges** on each **System.Data.DataTable** causes each **System.Data.DataRow** object's **System.Data.DataRow.AcceptChanges** method to be called. In this manner, you have multiple levels at which the method can be invoked. Calling the **System.Data.DataSet.AcceptChanges** of the **System.Data.DataSet** however, allows you to invoke the method on all subordinate objects (for example, tables and rows) with one call.

BeginInit

```

1
2 [C#] public void BeginInit();
3 [C++] public: __sealed void BeginInit();
4 [VB] NotOverridable Public Sub BeginInit()
5 [JScript] public function BeginInit();
6

```

#### 7 *Description*

8 Begins the initialization of a **System.Data.DataSet** that is used on a form  
 9 or used by another component. The initialization occurs at runtime.

10 The Visual Studio.NET design environment uses this method to start the  
 11 initialization of a component that is used on a form or used by another component.  
 12 The **System.Data.DataSet.EndInit** method ends the initialization. Using the  
 13 **BeginInit** and **EndInit** methods prevents the control from being used before it is  
 14 fully initialized.

15 Clear

```

16
17 [C#] public void Clear();
18 [C++] public: void Clear();
19 [VB] Public Sub Clear()
20 [JScript] public function Clear();
21

```

#### 22 *Description*

23 Clears the **System.Data.DataSet** of any data by removing all rows in all  
 24 tables.

25 Clone

```

1
2 [C#] public DataSet Clone();
3 [C++] public: DataSet* Clone();
4 [VB] Public Function Clone() As DataSet
5 [JScript] public function Clone() : DataSet;

```

#### *Description*

Clones the structure of the **System.Data.DataSet** , including all **System.Data.DataTable** schemas, relations, and constraints.

*Return Value:* A new **System.Data.DataSet** with the same schema as the current **System.Data.DataSet** .

If these classes have been subclassed, the clone will also be of the same subclasses.

#### *Copy*

```

16 [C#] public DataSet Copy();
17 [C++] public: DataSet* Copy();
18 [VB] Public Function Copy() As DataSet
19 [JScript] public function Copy() : DataSet;

```

#### *Description*

Copies both the structure and data for this **System.Data.DataSet** .

*Return Value:* A new **System.Data.DataSet** with the same structure (table schemas, relations, and constraints) and data as this **System.Data.DataSet** .

#### *EndInit*

```

1
2 [C#] public void EndInit();
3 [C++] public: __sealed void EndInit();
4 [VB] NotOverridable Public Sub EndInit()
5 [JScript] public function EndInit();
6

```

#### *Description*

Ends the initialization of a **System.Data.DataSet** that is used on a form or used by another component. The initialization occurs at runtime.

The Visual Studio.NET design environment uses this method to end the initialization of a component that is used on a form or used by another component. The **System.Data.DataSet.BeginInit** method starts the initialization. Using the **BeginInit** and **EndInit** methods prevents the control from being used before it is fully initialized.

#### *GetChanges*

```

17 [C#] public DataSet GetChanges();
18 [C++] public: DataSet* GetChanges();
19 [VB] Public Function GetChanges() As DataSet
20 [JScript] public function GetChanges() : DataSet; Gets a copy of the
21 System.Data.DataSet containing all changes made to it since it was last loaded,
22 or since System.Data.DataSet.AcceptChanges was called.
23

```

#### *Description*

Gets a copy of the **System.Data.DataSet** that contains all changes made to it since it was loaded or **System.Data.DataSet.AcceptChanges** was last called.

*Return Value:* A copy of the changes from this **System.Data.DataSet** that can have actions performed on it and subsequently be merged back in using **System.Data.DataSet.Merge(System.Data.DataSet)** , or **null** if none are found.

Gets a copy of the **System.Data.DataSet** that contains all changes made to it since it was loaded or **System.Data.DataSet.AcceptChanges** was last called.

This copy is particularly designed so that it can be merged back in to this original **System.Data.DataSet** . Relationship constraints may cause Unchanged parent rows to be included. If no rows of these rowStates are found, this method returns **null** .

GetChanges

[C#] public DataSet GetChanges(DataRowState rowStates);

[C++] public: DataSet\* GetChanges(DataRowState rowStates);

[VB] Public Function GetChanges(ByVal rowStates As DataRowState) As DataSet

[JScript] public function GetChanges(rowStates : DataRowState) : DataSet;

### Description

Gets a copy of the **System.Data.DataSet** containing all changes made to it since it was last loaded, or since **System.Data.DataSet.AcceptChanges** was called, filtered by **System.Data.DataRowState** .

*Return Value:* A filtered copy of the **System.Data.DataSet** that can have actions performed on it, and subsequently be merged back in using

1 **System.Data.DataSet.Merge(System.Data.DataSet)** . If no rows of the desired  
2 **System.Data.DataRowState** are found, the method returns **null** .

3 The **System.Data.DataSet.GetChanges** method is used to produce a  
4 second **System.Data.DataSet** object which contains only the changes introduced  
5 into the original. Use the *rowStates* argument to specify the type of changes the  
6 new object should include. One of the **System.Data.DataRowState** values.

7 **GetNestedChanges**

8  
9 [C#] public DataSet GetNestedChanges(DataRowState rowStates);

10 [C++] public: DataSet\* GetNestedChanges(DataRowState rowStates);

11 [VB] Public Function GetNestedChanges(ByVal rowStates As DataRowState) As

12 DataSet

13 [JScript] public function GetNestedChanges(rowStates : DataRowState) : DataSet;

14  
15 *Description*

16  
17 **GetSchemaSerializable**

18  
19 [C#] protected virtual XmlSchema GetSchemaSerializable();

20 [C++] protected: virtual XmlSchema\* GetSchemaSerializable();

21 [VB] Overridable Protected Function GetSchemaSerializable() As XmlSchema

22 [JScript] protected function GetSchemaSerializable() : XmlSchema;

23  
24 *Description*

Retrieves an **System.Xml.XmlTextReader** for the implementation of **IXmlSerializable** .

*Return Value:* An **System.Xml.XmlTextReader** .

A user should not call **System.Data.DataSet.GetSchemaSerializable** directly.

**GetSerializationData**

[C#] protected void GetSerializationData(SerializationInfo info, StreamingContext context);

[C++] protected: void GetSerializationData(SerializationInfo\* info, StreamingContext context);

[VB] Protected Sub GetSerializationData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function GetSerializationData(info : SerializationInfo, context : StreamingContext);

### *Description*

Retrieves **System.Runtime.Serialization.SerializationInfo** and **System.Runtime.Serialization.StreamingContext** information for the implementation of **IXmlSerializable** .

*Return Value:* **System.Runtime.Serialization.SerializationInfo** and **System.Runtime.Serialization.StreamingContext** information.

A user should not call **System.Data.DataSet.GetSerializationData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)** directly. The

data needed to serialize or deserialize an object. The source and destination of a given serialized stream.

### GetXml

```
[C#] public string GetXml();  
[C++] public: String* GetXml();  
[VB] Public Function GetXml() As String  
[JScript] public function GetXml() : String;
```

### *Description*

Returns the XML representation of the data stored in the **System.Data.DataSet**.

*Return Value:* String that is a representation of the data stored in the **System.Data.DataSet**.

If the **System.Data.DataSet** has changes, calling this method is identical to calling **System.Data.DataSet.WriteXml(System.IO.Stream)** with *XmlWriteMode* set to **DiffGram**; otherwise it is equivalent to calling **System.Data.DataSet.WriteXml(System.IO.Stream)** with *XmlWriteMode* set to **IgnoreSchema**.

### GetXmlSchema

```
[C#] public string GetXmlSchema();  
[C++] public: String* GetXmlSchema();  
[VB] Public Function GetXmlSchema() As String  
[JScript] public function GetXmlSchema() : String;
```

1  
2 *Description*

3 Returns the XSD schema for the XML representation of the data stored in  
4 the **System.Data.DataSet** .

5 *Return Value:* String that is the XSD schema for the XML representation of the  
6 data stored in the **System.Data.DataSet** .

7 Calling this method is identical to calling  
8 **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** , except that only  
9 the primary schema is written.

10 HasChanges

11  
12 [C#] public bool HasChanges();

13 [C++] public: bool HasChanges();

14 [VB] Public Function HasChanges() As Boolean

15 [JScript] public function HasChanges() : Boolean; Gets a value indicating whether  
16 the **System.Data.DataSet** has changes, including new, deleted, or modified rows.

17  
18 *Description*

19 Gets a value indicating whether the **System.Data.DataSet** has changes,  
20 including new, deleted, or modified rows.

21 *Return Value:* **true** , if the **System.Data.DataSet** has changes; otherwise, **false** .

22 HasChanges

23  
24 [C#] public bool HasChanges(DataRowState rowStates);

25 [C++] public: bool HasChanges(DataRowState rowStates);

1 [VB] Public Function HasChanges(ByVal rowStates As DataRowState) As  
2 Boolean

3 [JScript] public function HasChanges(rowStates : DataRowState) : Boolean;

4  
5 *Description*

6 Gets a value indicating whether the **System.Data.DataSet** has changes,  
7 including new, deleted, or modified rows, filtered by **System.Data.DataRowState**

8  
9 *Return Value:* **true** , if the **System.Data.DataSet** has changes; otherwise, **false** .

10 Examine the **System.Data.DataSet.HasChanges** property before invoking  
11 **System.Data.DataSet.GetChanges** method. One of the  
12 **System.Data.DataRowState** values.

13 InferXmlSchema

14  
15 [C#] public void InferXmlSchema(Stream stream, string[] nsArray);

16 [C++] public: void InferXmlSchema(Stream\* stream, String\* nsArray \_\_gc[]);

17 [VB] Public Sub InferXmlSchema(ByVal stream As Stream, ByVal nsArray() As  
18 String)

19 [JScript] public function InferXmlSchema(stream : Stream, nsArray : String[]);

20  
21 *Description*

22 Infers the XML schema from the specified **System.IO.TextReader** into the  
23 **System.Data.DataSet** . The **System.IO.Stream** from which to read. An array of  
24 namespace URI strings to be excluded from schema inference.

25 InferXmlSchema

```

1
2 [C#] public void InferXmlSchema(string fileName, string[] nsArray);
3 [C++] public: void InferXmlSchema(String* fileName, String* nsArray __gc[]);
4 [VB] Public Sub InferXmlSchema(ByVal fileName As String, ByVal nsArray()
5 As String)
6 [JScript] public function InferXmlSchema(fileName : String, nsArray : String[]);
7

```

### 8 *Description*

9        Infers the XML schema from the specified file into the  
10 **System.Data.DataSet** . The file name (including the path) from which to read. An  
11 array of namespace URI strings to be excluded from schema inference.

### 12 **InferXmlSchema**

```

13
14 [C#] public void InferXmlSchema(TextReader reader, string[] nsArray);
15 [C++] public: void InferXmlSchema(TextReader* reader, String* nsArray
16 __gc[]);
17 [VB] Public Sub InferXmlSchema(ByVal reader As TextReader, ByVal nsArray()
18 As String)
19 [JScript] public function InferXmlSchema(reader : TextReader, nsArray :
20 String[]);
21

```

### 22 *Description*

23        Infers the XML schema from the specified **System.IO.TextReader** into the  
24 **System.Data.DataSet** . The **System.IO.TextReader** from which to read. An  
25 array of namespace URI strings to be excluded from schema inference.

## InferXmlSchema

```
[C#] public void InferXmlSchema(XmlReader reader, string[] nsArray);  
[C++] public: void InferXmlSchema(XmlReader* reader, String* nsArray __gc[]);  
[VB] Public Sub InferXmlSchema(ByVal reader As XmlReader, ByVal nsArray()  
As String)  
[JScript] public function InferXmlSchema(reader : XmlReader, nsArray :  
String[]); Infers the XML schema from the specified System.IO.TextReader or  
file into the System.Data.DataSet.
```

### *Description*

Infer the XML schema from the specified **System.IO.TextReader** into the **System.Data.DataSet**. The **System.IO.TextReader** from which to read. An array of namespace URI strings to be excluded from schema inference.

### Merge

```
[C#] public void Merge(DataRow[] rows);  
[C++] public: void Merge(DataRow* rows[]);  
[VB] Public Sub Merge(ByVal rows() As DataRow)  
[JScript] public function Merge(rows : DataRow[]);
```

### *Description*

Merges this **System.Data.DataSet** with an array of **System.Data.DataRow** objects.

The **System.Data.DataSet.Merge(System.Data.DataSet)** method is used to merge two **System.Data.DataSet** objects that have largely similar schemas. A merge is typically used on a client application to incorporate the latest changes from a data source into an existing **System.Data.DataSet** . This allows the client application to have a refreshed **System.Data.DataSet** with the latest data from the data source. The array of **System.Data.DataRow** objects that will be merged into the **System.Data.DataSet**.

#### Merge

```
[C#] public void Merge(DataSet dataSet);  
[C++] public: void Merge(DataSet* dataSet);  
[VB] Public Sub Merge(ByVal dataSet As DataSet)  
[JScript] public function Merge(dataSet : DataSet); Merges this  
System.Data.DataSet with a specified System.Data.DataSet .
```

#### *Description*

Merges this **System.Data.DataSet** into a specified **System.Data.DataSet** .

The **System.Data.DataSet.Merge(System.Data.DataSet)** method is used to merge two **System.Data.DataSet** objects that have largely similar schemas. A merge is typically used on a client application to incorporate the latest changes from a data source into an existing **System.Data.DataSet** . This allows the client application to have a refreshed **System.Data.DataSet** with the latest data from the data source. The **System.Data.DataSet** whose data and schema will be merged.

#### Merge

```
[C#] public void Merge(DataTable table);
[C++] public: void Merge(DataTable* table);
[VB] Public Sub Merge(ByVal table As DataTable)
[JScript] public function Merge(table : DataTable);
```

### *Description*

Merges a **System.Data.DataSet** with a specified **System.Data.DataTable**.

The **System.Data.DataSet.Merge(System.Data.DataSet)** method is used to merge two **System.Data.DataSet** objects that have largely similar schemas. A merge is typically used on a client application to incorporate the latest changes from a data source into an existing **System.Data.DataSet**. This allows the client application to have a refreshed **System.Data.DataSet** with the latest data from the data source. The **System.Data.DataTable** whose data and schema will be merged.

### *Merge*

```
[C#] public void Merge(DataSet dataSet, bool preserveChanges);
[C++] public: void Merge(DataSet* dataSet, bool preserveChanges);
[VB] Public Sub Merge(ByVal dataSet As DataSet, ByVal preserveChanges As Boolean)
[JScript] public function Merge(dataSet : DataSet, preserveChanges : Boolean);
```

### *Description*

Merges this **System.Data.DataSet** with a specified **System.Data.DataSet** preserving changes according to the specified argument.

The **System.Data.DataSet.Merge(System.Data.DataSet)** method is used to merge two **System.Data.DataSet** objects that have largely similar schemas. A merge is typically used on a client application to incorporate the latest changes from a data source into an existing **System.Data.DataSet**. This allows the client application to have a refreshed **System.Data.DataSet** with the latest data from the data source. The **System.Data.DataSet** whose data and schema will be merged. A value indicating whether changes made to the current **System.Data.DataSet** should be maintained.

#### Merge

```
[C#] public void Merge(DataRow[] rows, bool preserveChanges,
MissingSchemaAction missingSchemaAction);
[C++] public: void Merge(DataRow* rows[], bool preserveChanges,
MissingSchemaAction missingSchemaAction);
[VB] Public Sub Merge(ByVal rows() As DataRow, ByVal preserveChanges As
Boolean, ByVal missingSchemaAction As MissingSchemaAction)
[JScript] public function Merge(rows : DataRow[], preserveChanges : Boolean,
missingSchemaAction : MissingSchemaAction);
```

#### *Description*

Merges this **System.Data.DataSet** with an array of **System.Data.DataRow** objects, preserving changes according to the specified argument, and handling an incompatible schema according to the specified argument.

The **System.Data.DataSet.Merge(System.Data.DataSet)** method is used to merge two **System.Data.DataSet** objects that have largely similar schemas. A

merge is typically used on a client application to incorporate the latest changes from a data source into an existing **System.Data.DataSet** . This allows the client application to have a refreshed **System.Data.DataSet** with the latest data from the data source. The array of **System.Data.DataRow** objects to merge with. **true** to preserve changes made to the **System.Data.DataSet**; otherwise, **false**. One of the **System.Data.MissingSchemaAction** values.

### Merge

```
[C#] public void Merge(DataSet dataSet, bool preserveChanges,
MissingSchemaAction missingSchemaAction);
[C++] public: void Merge(DataSet* dataSet, bool preserveChanges,
MissingSchemaAction missingSchemaAction);
[VB] Public Sub Merge(ByVal dataSet As DataSet, ByVal preserveChanges As
Boolean, ByVal missingSchemaAction As MissingSchemaAction)
[JScript] public function Merge(dataSet : DataSet, preserveChanges : Boolean,
missingSchemaAction : MissingSchemaAction);
```

### Description

Merges this **System.Data.DataSet** with a specified **System.Data.DataSet** preserving changes according to the specified argument, and handling an incompatible schema according to the specified argument.

The **System.Data.DataSet.Merge(System.Data.DataSet)** method is used to merge two **System.Data.DataSet** objects that have largely similar schemas. A merge is typically used on a client application to incorporate the latest changes from a data source into an existing **System.Data.DataSet** . This allows the client

application to have a refreshed **System.Data.DataSet** with the latest data from the data source. The **System.Data.DataSet** whose data and schema will be merged. A value indicating whether changes in the current (target) **System.Data.DataSet** should be maintained. One of the **System.Data.MissingSchemaAction** values.

### Merge

```
[C#] public void Merge(DataTable table, bool preserveChanges,
MissingSchemaAction missingSchemaAction);
[C++] public: void Merge(DataTable* table, bool preserveChanges,
MissingSchemaAction missingSchemaAction);
[VB] Public Sub Merge(ByVal table As DataTable, ByVal preserveChanges As
Boolean, ByVal missingSchemaAction As MissingSchemaAction)
[JScript] public function Merge(table : DataTable, preserveChanges : Boolean,
missingSchemaAction : MissingSchemaAction);
```

### Description

Merges this **System.Data.DataTable** with a specified **System.Data.DataTable** preserving changes according to the specified argument, and handling an incompatible schema according to the specified argument.

The **System.Data.DataSet.Merge(System.Data.DataSet)** method is used to merge two **System.Data.DataSet** objects that have largely similar schemas. A merge is typically used on a client application to incorporate the latest changes from a data source into an existing **System.Data.DataSet**. This allows the client application to have a refreshed **System.Data.DataSet** with the latest data from the data source. The **System.Data.DataSet** whose data and schema will be merged.

Whether changes in the current (target) **System.Data.DataSet** should be maintained. One of the **System.Data.MissingSchemaAction** values.

**OnPropertyChanging**

[C#] protected internal virtual void

**OnPropertyChanging**(PropertyChangedEventArgs pcevent);

[C++] protected public: virtual void

**OnPropertyChanging**(PropertyChangedEventArgs\* pcevent);

[VB] Overridable Protected Friend Dim Sub **OnPropertyChanging**(ByVal pcevent

As PropertyChangedEventArgs)

[JScript] package function **OnPropertyChanging**(pcevent :

PropertyChangedEventArgs);

#### *Description*

Raises the

**System.Data.DataSet.OnPropertyChanging(System.ComponentModel.PropertyChangedEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.ComponentModel.PropertyChangedEventArgs** that contains the event data.

**OnRemoveRelation**

[C#] protected virtual void **OnRemoveRelation**(DataRelation relation);

[C++] protected: virtual void **OnRemoveRelation**(DataRelation\* relation);

[VB] Overridable Protected Sub **OnRemoveRelation**(ByVal relation As

1 DataRelation)

2 [JScript] protected function OnRemoveRelation(relation : DataRelation);

3  
4 *Description*

5 This method should be overridden by subclasses to restrict tables being  
6 removed. The **System.Data.DataRelation** being removed.

7 OnRemoveTable

8  
9 [C#] protected virtual void OnRemoveTable(DataTable table);

10 [C++] protected: virtual void OnRemoveTable(DataTable\* table);

11 [VB] Overridable Protected Sub OnRemoveTable(ByVal table As DataTable)

12 [JScript] protected function OnRemoveTable(table : DataTable);

13  
14 *Description*

15 Occurs when when a **System.Data.DataTable** is being removed.

16 This method can be overridden by subclasses to restrict tables from being  
17 removed. The **System.Data.DataTable** being removed.

18 RaisePropertyChanging

19  
20 [C#] protected internal void RaisePropertyChanging(string name);

21 [C++] protected public: void RaisePropertyChanging(String\* name);

22 [VB] Protected Friend Dim Sub RaisePropertyChanging(ByVal name As String)

23 [JScript] package function RaisePropertyChanging(name : String);

24  
25 *Description*

Sends notification that the specified **System.Data.DataSet** property is about to change. The name of the property that is about to change.

### ReadXml

```
[C#] public XmlReadMode ReadXml(Stream stream);  
[C++] public: XmlReadMode ReadXml(Stream* stream);  
[VB] Public Function ReadXml(ByVal stream As Stream) As XmlReadMode  
[JScript] public function ReadXml(stream : Stream) : XmlReadMode;
```

### *Description*

Reads XML schema and data into the **System.Data.DataSet** using the specified **System.IO.Stream**.

Use the **System.Data.DataSet.ReadXml(System.Xml.XmlReader)** method to read an XML document that includes both schema and data. An object that derives from **System.IO.Stream**.

### ReadXml

```
[C#] public XmlReadMode ReadXml(string fileName);  
[C++] public: XmlReadMode ReadXml(String* fileName);  
[VB] Public Function ReadXml(ByVal fileName As String) As XmlReadMode  
[JScript] public function ReadXml(fileName : String) : XmlReadMode;
```

### *Description*

Reads XML schema and data into the **System.Data.DataSet** using the specified file.

Use the **System.Data.DataSet.ReadXml(System.Xml.XmlReader)** method to read an XML document that includes both schema and data. The file name (including the path) from which to read.

#### ReadXml

```
[C#] public XmlReadMode ReadXml(TextReader reader);  
[C++] public: XmlReadMode ReadXml(TextReader* reader);  
[VB] Public Function ReadXml(ByVal reader As TextReader) As XmlReadMode  
[JScript] public function ReadXml(reader : TextReader) : XmlReadMode;
```

#### *Description*

Reads XML schema and data into the **System.Data.DataSet** using the specified **System.IO.TextReader**.

Use the **System.Data.DataSet.ReadXml(System.Xml.XmlReader)** method to read an XML document that includes both schema and data. An object that derives from the **System.IO.TextReader** class.

#### ReadXml

```
[C#] public XmlReadMode ReadXml(XmlReader reader);  
[C++] public: XmlReadMode ReadXml(XmlReader* reader);  
[VB] Public Function ReadXml(ByVal reader As XmlReader) As XmlReadMode  
[JScript] public function ReadXml(reader : XmlReader) : XmlReadMode; Reads  
XML schema and data into the System.Data.DataSet.
```

#### *Description*

1 Reads XML schema and data into the **System.Data.DataSet** using the  
2 specified **System.Xml.XmlReader** .

3 Use the **System.Data.DataSet.ReadXml(System.Xml.XmlReader)**  
4 method to read an XML document that includes both schema and data. The  
5 **System.IO.TextReader** from which to read.

6 ReadXml

7  
8 [C#] public XmlReadMode ReadXml(Stream stream, XmlReadMode mode);  
9 [C++] public: XmlReadMode ReadXml(Stream\* stream, XmlReadMode mode);  
10 [VB] Public Function ReadXml(ByVal stream As Stream, ByVal mode As  
11 XmlReadMode) As XmlReadMode  
12 [JScript] public function ReadXml(stream : Stream, mode : XmlReadMode) :  
13 XmlReadMode;

14  
15 *Description*

16 Reads XML schema and data into the **System.Data.DataSet** using the  
17 specified **System.IO.Stream** and **System.Data.XmlReadMode** . The  
18 **System.IO.Stream** from which to read. One of the **System.Data.XmlReadMode**  
19 values.

20 ReadXml

21  
22 [C#] public XmlReadMode ReadXml(string fileName, XmlReadMode mode);  
23 [C++] public: XmlReadMode ReadXml(String\* fileName, XmlReadMode mode);  
24 [VB] Public Function ReadXml(ByVal fileName As String, ByVal mode As  
25 XmlReadMode) As XmlReadMode

```
1 [JScript] public function ReadXml(fileName : String, mode : XmlReadMode) :  
2 XmlReadMode;
```

#### 4 *Description*

5 Reads XML schema and data into the **System.Data.DataSet** using the  
6 specified file and **System.Data.XmlReadMode** . The file name (including the  
7 path) from which to read. One of the **System.Data.XmlReadMode** values.

#### 8 ReadXml

```
10 [C#] public XmlReadMode ReadXml(TextReader reader, XmlReadMode mode);
```

```
11 [C++] public: XmlReadMode ReadXml(TextReader* reader, XmlReadMode  
12 mode);
```

```
13 [VB] Public Function ReadXml(ByVal reader As TextReader, ByVal mode As  
14 XmlReadMode) As XmlReadMode
```

```
15 [JScript] public function ReadXml(reader : TextReader, mode : XmlReadMode) :  
16 XmlReadMode;
```

#### 18 *Description*

19 Reads XML schema and data into the **System.Data.DataSet** using the  
20 specified **System.IO.TextReader** and **System.Data.XmlReadMode** . The  
21 **System.IO.TextReader** from which to read. One of the  
22 **System.Data.XmlReadMode** values.

#### 23 ReadXml

```
25 [C#] public XmlReadMode ReadXml(XmlReader reader, XmlReadMode mode);
```

```

1  [C++] public: XmlReadMode ReadXml(XmlReader* reader, XmlReadMode
2  mode);
3  [VB] Public Function ReadXml(ByVal reader As XmlReader, ByVal mode As
4  XmlReadMode) As XmlReadMode
5  [JScript] public function ReadXml(reader : XmlReader, mode : XmlReadMode) :
6  XmlReadMode; Writes the current schema and data for the System.Data.DataSet
7  to an XML document using the specified System.Data.XmlReadMode .
8

```

### *Description*

Writes schema and data for the DataSet. The **System.IO.TextReader** from which to read. One of the **System.Data.XmlReadMode** values.

### **ReadXmlSchema**

```

14 [C#] public void ReadXmlSchema(Stream stream);
15 [C++] public: void ReadXmlSchema(Stream* stream);
16 [VB] Public Sub ReadXmlSchema(ByVal stream As Stream)
17 [JScript] public function ReadXmlSchema(stream : Stream);
18

```

### *Description*

Reads the XML schema from the specified **System.IO.Stream** into the **System.Data.DataSet** .

Use the

**System.Data.DataSet.ReadXmlSchema(System.Xml.XmlReader)** method to create the schema for a **System.Data.DataSet** . The schema includes table, relation, and constraint definitions. To write a schema to an XML document, use



## Description

Reads the XML schema from the specified **System.IO.TextReader** into the **System.Data.DataSet** .

Use the **System.Data.DataSet.ReadXmlSchema(System.Xml.XmlReader)** method to create the schema for a **System.Data.DataSet** . The schema includes table, relation, and constraint definitions. To write a schema to an XML document, use the **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** method. The **System.IO.TextReader** from which to read.

### ReadXmlSchema

[C#] public void ReadXmlSchema(XmlReader reader);

[C++] public: void ReadXmlSchema(XmlReader\* reader);

[VB] Public Sub ReadXmlSchema(ByVal reader As XmlReader)

[JScript] public function ReadXmlSchema(reader : XmlReader); Reads an XML schema into the **System.Data.DataSet** .

## Description

Reads the XML schema from the specified **System.Xml.XmlReader** into the **System.Data.DataSet** .

Use the **System.Data.DataSet.ReadXmlSchema(System.Xml.XmlReader)** method to create the schema for a **System.Data.DataSet** . The schema includes table,

relation, and constraint definitions. The **System.Xml.XmlReader** from which to read.

### ReadXmlSerializable

[C#] protected virtual void ReadXmlSerializable(XmlReader reader);  
 [C++] protected: virtual void ReadXmlSerializable(XmlReader\* reader);  
 [VB] Overridable Protected Sub ReadXmlSerializable(ByVal reader As XmlReader)  
 [JScript] protected function ReadXmlSerializable(reader : XmlReader);

#### *Description*

Reads XML serialization information for the implementation of **IXmlSerializable**.

*Return Value:* An **System.Xml.XmlTextReader** object.

A user should not call **System.Data.DataSet.ReadXmlSerializable(System.Xml.XmlReader)** directly. The **System.Xml.XmlTextReader** object.

### RejectChanges

[C#] public virtual void RejectChanges();  
 [C++] public: virtual void RejectChanges();  
 [VB] Overridable Public Sub RejectChanges()  
 [JScript] public function RejectChanges();

#### *Description*

Rolls back all the changes made to the **System.Data.DataSet** since it was created, or since the last time **System.Data.DataSet.AcceptChanges** was called.

Invoke the **System.Data.DataSet.RejectChanges** to call the **System.Data.DataTable.RejectChanges** method on all **System.Data.DataTable** objects contained by the **System.Data.DataSet** . Additionally, any **System.Data.Constraint** rules contained by the **System.Data.DataSet** are enforced.

## Reset

[C#] public virtual void Reset();

[C++] public: virtual void Reset();

[VB] Overridable Public Sub Reset()

[JScript] public function Reset();

## Description

Resets the **System.Data.DataSet** to its original state. Subclasses should override **System.Data.DataSet.Reset** to restore a **System.Data.DataSet** to its original state.

## ShouldSerializeRelations

[C#] protected virtual bool ShouldSerializeRelations();

[C++] protected: virtual bool ShouldSerializeRelations();

[VB] Overridable Protected Function ShouldSerializeRelations() As Boolean

[JScript] protected function ShouldSerializeRelations() : Boolean;

## Description

Gets a value indicating whether **System.Data.DataSet.Relations** property should be persisted.

**Return Value:** **true** if the property value has been changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Data.DataSet** , or creating your own control incorporating the **System.Data.DataSet** .

**ShouldSerializeTables**

[C#] protected virtual bool ShouldSerializeTables();

[C++] protected: virtual bool ShouldSerializeTables();

[VB] Overridable Protected Function ShouldSerializeTables() As Boolean

[JScript] protected function ShouldSerializeTables() : Boolean;

## Description

Gets a value indicating whether **System.Data.DataSet.Tables** property should be persisted.

**Return Value:** **true** if the property value has been changed from its default; otherwise, **false** .

You typically use this method only if you are either creating a designer for the **System.Data.DataSet** , or creating your own control incorporating the **System.Data.DataSet** .

**IListSource.GetList**

```

1
2 [C#] IList IListSource.GetList();
3 [C++] IList* IListSource::GetList();
4 [VB] Function GetList() As IList Implements IListSource.GetList
5 [JScript] function IListSource.GetList() : IList;
6     ISerializable.GetObjectData
7
8 [C#] void ISerializable.GetObjectData(SerializationInfo info, StreamingContext
9 context);
10 [C++] void ISerializable::GetObjectData(SerializationInfo* info,
11 StreamingContext context);
12 [VB] Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As
13 StreamingContext) Implements ISerializable.GetObjectData
14 [JScript] function ISerializable.GetObjectData(info : SerializationInfo, context :
15 StreamingContext);
16     IXmlSerializable.GetSchema
17
18 [C#] XmlSchema IXmlSerializable.GetSchema();
19 [C++] XmlSchema* IXmlSerializable::GetSchema();
20 [VB] Function GetSchema() As XmlSchema Implements
21 IXmlSerializable.GetSchema
22 [JScript] function IXmlSerializable.GetSchema() : XmlSchema;
23     IXmlSerializable.ReadXml
24
25 [C#] void IXmlSerializable.ReadXml(XmlReader reader);

```

```

1  [C++] void IXmlSerializable::ReadXml(XmlReader* reader);
2  [VB] Sub ReadXml(ByVal reader As XmlReader) Implements
3  IXmlSerializable.ReadXml
4  [JScript] function IXmlSerializable.ReadXml(reader : XmlReader);
5      IXmlSerializable.WriteXml
6
7  [C#] void IXmlSerializable.WriteXml(XmlWriter writer);
8  [C++] void IXmlSerializable::WriteXml(XmlWriter* writer);
9  [VB] Sub WriteXml(ByVal writer As XmlWriter) Implements
10 IXmlSerializable.WriteXml
11 [JScript] function IXmlSerializable.WriteXml(writer : XmlWriter);
12     WriteXml
13
14 [C#] public void WriteXml(Stream stream);
15 [C++] public: void WriteXml(Stream* stream);
16 [VB] Public Sub WriteXml(ByVal stream As Stream)
17 [JScript] public function WriteXml(stream : Stream); Writes XML schema and
18 data from the System.Data.DataSet .
19
20 Description
21     Writes the current schema and data for the System.Data.DataSet using the
22 specified System.IO.Stream .
23     Use the System.Data.DataSet.WriteXml(System.IO.Stream) method to
24 write an XML document that includes both schema and data of a
25 System.Data.DataSet . To read an XML document, that includes schema and

```



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public void WriteXml(XmlWriter writer);
[C++] public: void WriteXml(XmlWriter* writer);
[VB] Public Sub WriteXml(ByVal writer As XmlWriter)
[JScript] public function WriteXml(writer : XmlWriter);
```

*Description*

Writes the current schema and data for the **System.Data.DataSet** to the specified **System.Xml.XmlWriter** . The **System.Xml.XmlWriter** with which to write.

**WriteXml**

```
[C#] public void WriteXml(Stream stream, XmlWriteMode mode);
[C++] public: void WriteXml(Stream* stream, XmlWriteMode mode);
[VB] Public Sub WriteXml(ByVal stream As Stream, ByVal mode As
XmlWriteMode)
[JScript] public function WriteXml(stream : Stream, mode : XmlWriteMode);
```

*Description*

Writes the current schema and data for the **System.Data.DataSet** using the specified **System.IO.Stream** and **System.Data.XmlWriteMode** . A **System.IO.Stream** object used to write to a file. One of the **System.Data.XmlWriteMode** values.

**WriteXml**

```

1
2 [C#] public void WriteXml(string fileName, XmlWriteMode mode);
3 [C++] public: void WriteXml(String* fileName, XmlWriteMode mode);
4 [VB] Public Sub WriteXml(ByVal fileName As String, ByVal mode As
5 XmlWriteMode)
6 [JScript] public function WriteXml(fileName : String, mode : XmlWriteMode);
7

```

### *Description*

Writes the current schema and data for the **System.Data.DataSet** to the specified file using the specified **System.Data.XmlWriteMode**.

Use the **System.Data.DataSet.WriteXml(System.IO.Stream)** method to write an XML document that includes both schema and data of a **System.Data.DataSet**. To read an XML document, that includes schema and data, use the **System.Data.DataSet.ReadXml(System.Xml.XmlReader)** method.

The file name (including the path) to which to write. One of the **System.Data.XmlWriteMode** values.

### **WriteXml**

```

17
18
19 [C#] public void WriteXml(TextWriter writer, XmlWriteMode mode);
20 [C++] public: void WriteXml(TextWriter* writer, XmlWriteMode mode);
21 [VB] Public Sub WriteXml(ByVal writer As TextWriter, ByVal mode As
22 XmlWriteMode)
23 [JScript] public function WriteXml(writer : TextWriter, mode : XmlWriteMode);
24

```

### *Description*

Writes the current schema and data for the **System.Data.DataSet** using the specified **System.IO.TextWriter** and **System.Data.XmlWriteMode** .

Use the **System.Data.DataSet.WriteXml(System.IO.Stream)** method to write an XML document that includes both schema and data of a **System.Data.DataSet** . To read an XML document, that includes schema and data, use the **System.Data.DataSet.ReadXml(System.Xml.XmlReader)** method. A **System.IO.TextWriter** object used to write the document. One of the **System.Data.XmlWriteMode** values.

#### WriteXml

```
[C#] public void WriteXml(XmlWriter writer, XmlWriteMode mode);
[C++] public: void WriteXml(XmlWriter* writer, XmlWriteMode mode);
[VB] Public Sub WriteXml(ByVal writer As XmlWriter, ByVal mode As
XmlWriteMode)
[JScript] public function WriteXml(writer : XmlWriter, mode : XmlWriteMode);
```

#### *Description*

Writes the current schema and data for the **System.Data.DataSet** using the specified **System.Xml.XmlWriter** and **System.Data.XmlWriteMode** .

Use the **System.Data.DataSet.WriteXml(System.IO.Stream)** method to write an XML document that includes both schema and data of a **System.Data.DataSet** . To read an XML document, that includes schema and data, use the **System.Data.DataSet.ReadXml(System.Xml.XmlReader)** method. The **System.Xml.XmlWriter** with which to write. One of the **System.Data.XmlWriteMode** values.

## WriteXmlSchema

[C#] public void WriteXmlSchema(Stream stream);  
[C++] public: void WriteXmlSchema(Stream\* stream);  
[VB] Public Sub WriteXmlSchema(ByVal stream As Stream)  
[JScript] public function WriteXmlSchema(stream : Stream); Writes the  
**System.Data.DataSet** structure as an XML schema.

### *Description*

Writes the **System.Data.DataSet** structure as an XML schema to using the specified **System.IO.Stream** object.

Use the **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** method to write the schema for a **System.Data.DataSet** to an XML document. The schema includes table, relation, and constraint definitions. To write a schema to an XML document, use the **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** method. A **System.IO.Stream** object used to write to a file.

## WriteXmlSchema

[C#] public void WriteXmlSchema(string fileName);  
[C++] public: void WriteXmlSchema(String\* fileName);  
[VB] Public Sub WriteXmlSchema(ByVal fileName As String)  
[JScript] public function WriteXmlSchema(fileName : String);

### *Description*

1 Writes the **System.Data.DataSet** structure as an XML schema to a file.

2 Use the **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)**

3 method to write the schema for a **System.Data.DataSet** to an XML document.

4 The schema includes table, relation, and constraint definitions. To write a schema

5 to an XML document, use the

6 **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** method. The file

7 name (including the path) to which to write.

8 **WriteXmlSchema**

9

10 [C#] public void WriteXmlSchema(TextWriter writer);

11 [C++] public: void WriteXmlSchema(TextWriter\* writer);

12 [VB] Public Sub WriteXmlSchema(ByVal writer As TextWriter)

13 [JScript] public function WriteXmlSchema(writer : TextWriter);

14

15 *Description*

16 Writes the **System.Data.DataSet** structure as an XML schema to a

17 **System.IO.TextWriter** object.

18 Use the **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)**

19 method to write the schema for a **System.Data.DataSet** to an XML document.

20 The schema includes table, relation, and constraint definitions. To write a schema

21 to an XML document, use the

22 **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** method. The

23 **System.IO.TextWriter** object with which to write.

24 **WriteXmlSchema**

25

```

1 [C#] public void WriteXmlSchema(XmlWriter writer);
2
3 [C++] public: void WriteXmlSchema(XmlWriter* writer);
4
5 [VB] Public Sub WriteXmlSchema(ByVal writer As XmlWriter)
6
7 [JScript] public function WriteXmlSchema(writer : XmlWriter);
8

```

### *Description*

Writes the **System.Data.DataSet** structure as an XML schema to an **System.Xml.XmlWriter** object.

Use the **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** method to write the schema for a **System.Data.DataSet** to an XML document. The schema includes table, relation, and constraint definitions. To write a schema to an XML document, use the **System.Data.DataSet.WriteXmlSchema(System.IO.Stream)** method. The **System.Xml.XmlWriter** with which to write.

DataSysDescriptionAttribute class (System.Data)

WriteXmlSchema

### *Description*

DescriptionAttribute marks a property, event, or extender with a description. Visual designers can display this description when referencing the member.

DataSysDescriptionAttribute

*Example Syntax:*

## WriteXmlSchema

```
[C#] public DataSysDescriptionAttribute(string description);  
[C++] public: DataSysDescriptionAttribute(String* description);  
[VB] Public Sub New(ByVal description As String)  
[JScript] public function DataSysDescriptionAttribute(description : String);
```

### *Description*

Constructs a new sys description. description text.

Description

WriteXmlSchema

```
[C#] public override string Description {get;}  
[C++] public: __property virtual String* get_Description();  
[VB] Overrides Public ReadOnly Property Description As String  
[JScript] public function get Description() : String;
```

### *Description*

Retrieves the description text.

*Return Value:* description Retrieves the description text.

DescriptionValue

TypeId

DataTable class (System.Data)

ToString

1  
2  
3 *Description*

4 Represents one table of in-memory data.

5 The **System.Data.DataTable** is a central object in the ADO.NET library.

6 Other objects that use the **System.Data.DataTable** include the

7 **System.Data.DataSet** and the **System.Data.DataView** .

8 ToString

9  
10 [C#] protected internal bool fInitInProgress;

11 [C++] protected public: bool fInitInProgress;

12 [VB] Internal fInitInProgress As Boolean

13 [JScript] package var fInitInProgress : Boolean;

14  
15 *Description*

16  
17 DataTable

18 *Example Syntax:*

19 ToString

20  
21 [C#] public DataTable();

22 [C++] public: DataTable();

23 [VB] Public Sub New()

24 [JScript] public function DataTable(); Initializes a new instance of the

25 **System.Data.DataTable** class.

1  
2 *Description*

3        Initializes a new instance of the **System.Data.DataTable** class with no  
4 arguments.

5        The constructor sets initial values for all properties of the  
6 **System.Data.DataTable** object. The following table shows the properties and  
7 their default values. When an instance **System.Data.DataTable** is created, the  
8 following read/write properties are set to initial values.

9        DataTable

10       *Example Syntax:*

11       ToString

12  
13 [C#] public DataTable(string tableName);

14 [C++] public: DataTable(String\* tableName);

15 [VB] Public Sub New(ByVal tableName As String)

16 [JScript] public function DataTable(tableName : String);

17  
18 *Description*

19        Intitalizes a new instance of the **System.Data.DataTable** class with the  
20 specified table name. The name to give the table. If **null** or an empty string, a  
21 default name will be given when added to the **System.Data.DataTableCollection**.

22        DataTable

23       *Example Syntax:*

24       ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] protected DataTable(SerializationInfo info, StreamingContext context);
[C++] protected: DataTable(SerializationInfo* info, StreamingContext context);
[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
StreamingContext)
[JScript] protected function DataTable(info : SerializationInfo, context :
StreamingContext);
```

*Description*

Initializes a new instance of the **System.Data.DataTable** class with the **System.Runtime.Serialization.SerializationInfo** and the **System.Runtime.Serialization.StreamingContext** .

This implemenation of the **System.Data.DataTable** constructor is required for **System.Runtime.Serialization.ISerializable** . The data needed to serialize or deserialize an object. The source and destination of a given serialized stream.

CaseSensitive  
ToString

```
[C#] public bool CaseSensitive {get; set;}
[C++] public: __property bool get_CaseSensitive();public: __property void
set_CaseSensitive(bool);
[VB] Public Property CaseSensitive As Boolean
[JScript] public function get CaseSensitive() : Boolean;public function set
CaseSensitive(Boolean);
```

## Description

Indicates whether string comparisons within the table are case-sensitive.

The **System.Data.DataTable.CaseSensitive** property affects string comparisons in sorting, searching, and filtering.

ChildRelations

ToString

[C#] public DataRelationCollection ChildRelations {get;}

[C++] public: \_\_property DataRelationCollection\* get\_ChildRelations();

[VB] Public ReadOnly Property ChildRelations As DataRelationCollection

[JScript] public function get ChildRelations() : DataRelationCollection;

## Description

Gets the collection of child relations for this **System.Data.DataTable**.

A **System.Data.DataRelation** defines the relationship between two tables.

Typically, two tables are linked through a single field that contains the same data.

For example, a table which contains address data may have a single field containing codes that represent countries/regions. A second table that contains country/region data will have a single field that contains the code that identifies the country/region, and it is this code which is inserted into the corresponding field in the first table. A **System.Data.DataRelation**, then, contains at least four pieces of information: (1) the name of the first table, (2) the column name in the first table, (3) the name of the second table, and (4) the column name in the second table.

Columns

ToString

[C#] public DataColumnCollection Columns {get;}

[C++] public: \_\_property DataColumnCollection\* get\_Columns();

[VB] Public ReadOnly Property Columns As DataColumnCollection

[JScript] public function get Columns() : DataColumnCollection;

*Description*

Gets the collection of columns that belong to this table.

The **System.Data.DataColumnCollection** determines the schema of a table by defining the data type of each column.

Constraints

ToString

[C#] public ConstraintCollection Constraints {get;}

[C++] public: \_\_property ConstraintCollection\* get\_Constraints();

[VB] Public ReadOnly Property Constraints As ConstraintCollection

[JScript] public function get Constraints() : ConstraintCollection;

*Description*

Gets the collection of constraints maintained by this table.

A **System.Data.ForeignKeyConstraint** restricts the action performed when a value in a column (or columns) is either deleted or updated. Such a constraint is intended to be used with primary key columns. In a parent/child

relationship between two tables, deleting a value from the parent table can affect the child rows in one of the following ways.

Container

DataSet

ToString

#### *Description*

Gets the **System.Data.DataSet** that this table belongs to.

If a control is data bound to a **System.Data.DataTable** , and the table belongs to a **System.Data.DataSet** , you can get to the **System.Data.DataSet** through this property.

DefaultView

ToString

[C#] public DataView DefaultView {get;}

[C++] public: \_\_property DataView\* get\_DefaultView();

[VB] Public ReadOnly Property DefaultView As DataView

[JScript] public function get DefaultView() : DataView;

#### *Description*

Gets a customized view of the table which may include a filtered view, or a cursor position.

The **System.Data.DataTable.DefaultView** property returns a **System.Data.DataView** you can use to sort, filter, and search a **System.Data.DataTable**.

DesignMode  
DisplayExpression  
ToString

#### *Description*

Gets or sets the expression that will return a value used to represent this table in the user interface.

For rules on forming the expression string, see the **System.Data.DataColumn.Expression** property.

Events  
ExtendedProperties  
ToString

#### *Description*

Gets the collection of customized user information.

Use the **System.Data.DataTable.ExtendedProperties** to add custom information to a **System.Data.DataTable**. Add information with the **Add** method. Retrieve information with the **Item** method.

HasErrors  
ToString

1 [C#] public bool HasErrors {get;}

2 [C++] public: \_\_property bool get\_HasErrors();

3 [VB] Public ReadOnly Property HasErrors As Boolean

4 [JScript] public function get HasErrors() : Boolean;

5  
6  
7 *Description*

8 Gets a value indicating whether there are errors in any of the rows in any of  
9 the tables of the **System.Data.DataSet** to which the table belongs.

10 As users work on a set of data contained in a **System.Data.DataSet** , you  
11 can mark each change with an error if the change causes some validation failure.  
12 You can mark an entire **System.Data.DataRow** with an error message using the  
13 **System.Data.DataRow.RowError** property. You can also set errors on each  
14 column of the row with the  
15 **System.Data.DataRow.SetColumnError(System.Int32,System.String)** method.

16 Locale

17 ToString

18  
19 [C#] public CultureInfo Locale {get; set;}

20 [C++] public: \_\_property CultureInfo\* get\_Locale();public: \_\_property void  
21 set\_Locale(CultureInfo\*);

22 [VB] Public Property Locale As CultureInfo

23 [JScript] public function get Locale() : CultureInfo;public function set  
24 Locale(CultureInfo);

## *Description*

Gets or sets the locale information used to compare strings within the table.

A **System.Globalization.CultureInfo** represents the software preferences of a particular culture or community.

MinimumCapacity

ToString

[C#] public int MinimumCapacity {get; set;}

[C++] public: \_\_property int get\_MinimumCapacity();public: \_\_property void set\_MinimumCapacity(int);

[VB] Public Property MinimumCapacity As Integer

[JScript] public function get MinimumCapacity() : int;public function set MinimumCapacity(int);

## *Description*

Gets or sets the initial starting size for this table.

The **System.Data.DataTable.MinimumCapacity** allows the system to create an appropriate set of resources before fetching data. In a situation when performance is critical, setting this property can optimize performance.

Namespace

ToString

[C#] public string Namespace {get; set;}

[C++] public: \_\_property String\* get\_Namespace();public: \_\_property void

```

1 set_Namespace(String*);
2 [VB] Public Property Namespace As String
3 [JScript] public function get Namespace() : String;public function set
4 Namespace(String);
5

```

#### *Description*

Gets or sets the namespace for the XML representation of the data stored in the **System.Data.DataTable** .

ParentRelations

ToString

```

12 [C#] public DataRelationCollection ParentRelations {get;}
13 [C++] public: __property DataRelationCollection* get _ParentRelations();
14 [VB] Public ReadOnly Property ParentRelations As DataRelationCollection
15 [JScript] public function get ParentRelations() : DataRelationCollection;
16

```

#### *Description*

Gets the collection of parent relations for this **System.Data.DataTable** .

Prefix

ToString

```

22 [C#] public string Prefix {get; set;}
23 [C++] public: __property String* get _Prefix();public: __property void
24 set _Prefix(String*);
25 [VB] Public Property Prefix As String

```

[JScript] public function get Prefix() : String;public function set Prefix(String);

### *Description*

Gets or sets the namespace for the XML representation of the data stored in the **System.Data.DataTable** .

PrimaryKey

ToString

[C#] public DataColumn[] PrimaryKey {get; set;}

[C++] public: \_\_property DataColumn\* get\_PrimaryKey();public: \_\_property

void set\_PrimaryKey(DataColumn\*[]);

[VB] Public Property PrimaryKey As DataColumn ()

[JScript] public function get PrimaryKey() : DataColumn[];public function set

PrimaryKey(DataColumn[]);

### *Description*

Gets or sets an array of columns that function as primary keys for the data table.

The primary key of a table must be unique to identify the record in the table. It's also possible to have a table with a primary key made up of two or more columns. This occurs when a single column can't contain enough unique values. For example, a two column primary key might consist of a "FirstName" and "LastName" column. Because primary keys can be made up of more than one column, the **System.Data.DataTable.PrimaryKey** property consists of an array of **System.Data.DataColumn** objects.

Rows

ToString

[C#] public DataRowCollection Rows {get;}

[C++] public: \_\_property DataRowCollection\* get\_Rows();

[VB] Public ReadOnly Property Rows As DataRowCollection

[JScript] public function get Rows() : DataRowCollection;

### *Description*

Gets the collection of rows that belong to this table.

To create a new **System.Data.DataRow**, you must use the **System.Data.DataTable.NewRow** method to return a new object. Such an object is automatically configured with according to the schema defined for the **System.Data.DataTable** through its collection of **System.Data.DataColumn** objects. After creating a new row and setting the values for each column in the row, add the row to the **DataRowCollection** using the **Add** method.

Site

ToString

[C#] public override ISite Site {get; set;}

[C++] public: \_\_property virtual ISite\* get\_Site();public: \_\_property virtual void set\_Site(ISite\*);

[VB] Overrides Public Property Site As ISite

[JScript] public function get Site() : ISite;public function set Site(ISite);

1  
2 *Description*

3 Gets or sets an **System.ComponentModel.ISite** for the  
4 **System.Data.DataTable** .

5 Sites bind a **System.ComponentModel.Component** to a  
6 **System.ComponentModel.Container** and enable communication between them,  
7 as well as provide a way for the container to manage its components.

8 **TableName**

9 **ToString**

10  
11 [C#] public string TableName {get; set;}

12 [C++] public: \_\_property String\* get\_TableName();public: \_\_property void  
13 set\_TableName(String\*);

14 [VB] Public Property TableName As String

15 [JScript] public function get TableName() : String;public function set  
16 TableName(String);

17  
18 *Description*

19 Gets or sets the name of the the **System.Data.DataTable** .

20 The **System.Data.DataTable.TableName** is used to return this table from  
21 the parent **System.Data.DataSet** object's **System.Data.DataTableCollection**  
22 (returned by the **System.Data.DataSet.Tables** property).

23 **ToString**

24  
25 [C#] public event DataColumnChangeEventHandler ColumnChanged;

[C++] public: \_\_event DataColumnChangeEventHandler\* ColumnChanged;

[VB] Public Event ColumnChanged As DataColumnChangeEventHandler

*Description*

Occurs when after a value has been changed for the specified

**System.Data.DataColumn** in a **System.Data.DataRow** .

ToString

[C#] public event DataColumnChangeEventHandler ColumnChanging;

[C++] public: \_\_event DataColumnChangeEventHandler\* ColumnChanging;

[VB] Public Event ColumnChanging As DataColumnChangeEventHandler

*Description*

Occurs when a value is being changed for the specified

**System.Data.DataColumn** in a **System.Data.DataRow** .

ToString

*Description*

Occurs after a **System.Data.DataRow** has been changed successfully.

ToString

[C#] public event DataRowChangeEventHandler RowChanging;

[C++] public: \_\_event DataRowChangeEventHandler\* RowChanging;

[VB] Public Event RowChanging As DataRowChangeEventHandler

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Occurs when a **System.Data.DataRow** is changing.

ToString

```
[C#] public event DataRowChangeEventHandler RowDeleted;
[C++] public: __event DataRowChangeEventHandler* RowDeleted;
[VB] Public Event RowDeleted As DataRowChangeEventHandler
```

*Description*

Occurs after a row in the table has been deleted.

ToString

```
[C#] public event DataRowChangeEventHandler RowDeleting;
[C++] public: __event DataRowChangeEventHandler* RowDeleting;
[VB] Public Event RowDeleting As DataRowChangeEventHandler
```

*Description*

Occurs before a row in the table is about to be deleted.

AcceptChanges

```
[C#] public void AcceptChanges();
[C++] public: void AcceptChanges();
[VB] Public Sub AcceptChanges()
[JavaScript] public function AcceptChanges();
```

## *Description*

Commits all the changes made to this table since the last time **System.Data.DataTable.AcceptChanges** was called.

When **System.Data.DataTable.AcceptChanges** is called, any **System.Data.DataRow** object still in edit mode successfully ends its edits. The **System.Data.DataRowState** also changes: all **Added** and **Modified** rows become **Unchanged** ; **Deleted** rows are removed.

### BeginInit

```
[C#] public void BeginInit();
[C++] public: __sealed void BeginInit();
[VB] NotOverridable Public Sub BeginInit()
[JScript] public function BeginInit();
```

## *Description*

Begins the initialization of a **System.Data.DataTable** that is used on a form or used by another component. The initialization occurs at runtime.

The Visual Studio.NET design environment uses this method to start the initialization of a component that is used on a form or used by another component. The **System.Data.DataTable.EndInit** method ends the initialization. Using the **BeginInit** and **EndInit** methods prevents the control from being used before it is fully initialized.

### BeginLoadData

```
[C#] public void BeginLoadData();
[C++] public: void BeginLoadData();
[VB] Public Sub BeginLoadData()
[JScript] public function BeginLoadData();
```

#### *Description*

Turns off notifications, index maintenance, and constraints while loading data.

Use **System.Data.DataTable.BeginLoadData** in conjunction with **System.Data.DataTable.LoadDataRow(System.Object[],System.Boolean)** and **System.Data.DataTable.EndLoadData**.

#### Clear

```
[C#] public void Clear();
[C++] public: void Clear();
[VB] Public Sub Clear()
[JScript] public function Clear();
```

#### *Description*

Clears the **System.Data.DataTable** of all data.

All rows in all tables are removed. An exception is generated if the table has any enforced child relations that would cause child rows to be stranded.

#### Clone

```

1
2 [C#] public DataTable Clone();
3 [C++] public: DataTable* Clone();
4 [VB] Public Function Clone() As DataTable
5 [JScript] public function Clone() : DataTable;

```

#### 6 *Description*

8 Clones the structure of the **System.Data.DataTable** , including all  
9 **System.Data.DataTable** schemas, relations, and constraints.

10 *Return Value:* A new **System.Data.DataTable** with the same schema as the  
11 current **System.Data.DataTable** .

12 If these classes have been subclassed, the clone will also be of the same  
13 subclasses.

#### 14 **Compute**

```

15
16 [C#] public object Compute(string expression, string filter);
17 [C++] public: Object* Compute(String* expression, String* filter);
18 [VB] Public Function Compute(ByVal expression As String, ByVal filter As
19 String) As Object
20 [JScript] public function Compute(expression : String, filter : String) : Object;

```

#### 21 *Description*

22 Computes the given expression on the current rows that pass the filter  
23 criteria.

24 *Return Value:* An **System.Object** , set to the result of the computation.

The *expression* parameter requires an aggregate function. For example, the following is a legal expression: Count(Quantity) But this expression is not: Sum (Quantity \* UnitPrice) If you must perform an operation on two or more columns, you should create a **System.Data.DataColumn** , set its **System.Data.DataColumn.Expression** property to an appropriate expression, and use an aggregate expression on the resulting column. In that case, given a **System.Data.DataColumn** with the name "total," and the **System.Data.DataColumn.Expression** property set to: "Quantity \* UnitPrice" The expression argument for the **System.Data.DataTable.Compute(System.String,System.String)** method would then be: Sum(total) The second parameter *filter* determines which rows are used in the expression. For example, if the table contains a date column named "colDate", you could limit the rows with the following expression: colDate > 1/1/99 AND colDate < 17/1/99 For rules on creating expressions for both parameters, see the **System.Data.DataColumn.Expression** property of the **System.Data.DataColumn** class. The expression to compute. The filter to limit the rows that evaluate in the expression.

#### Copy

```
[C#] public DataTable Copy();
[C++] public: DataTable* Copy();
[VB] Public Function Copy() As DataTable
[JavaScript] public function Copy() : DataTable;
```

#### Description

Copies both the structure and data for this **System.Data.DataTable** .

*Return Value:* A new **System.Data.DataTable** with the same structure (table schemas, relations, and constraints) and data as this **System.Data.DataTable** .

**EndInit**

[C#] public void EndInit();

[C++] public: \_\_sealed void EndInit();

[VB] NotOverridable Public Sub EndInit()

[JScript] public function EndInit();

### *Description*

Ends the initialization of a **System.Data.DataTable** that is used on a form or used by another component. The initialization occurs at runtime.

The Visual Studio.NET design environment uses this method to end the initialization of a component that is used on a form or used by another component. The **System.Data.DataTable.BeginInit** method starts the initialization. Using the **BeginInit** and **EndInit** methods prevents the control from being used before it is fully initialized.

**EndLoadData**

[C#] public void EndLoadData();

[C++] public: void EndLoadData();

[VB] Public Sub EndLoadData()

[JScript] public function EndLoadData();

1  
2 *Description*

3 Turns off notifications, index maintenance, and constraints while loading  
4 data.

5 Use **System.Data.DataTable.EndLoadData** in conjunction with  
6 **System.Data.DataTable.LoadDataRow(System.Object[],System.Boolean)** and  
7 **System.Data.DataTable.BeginLoadData** .

8 **GetChanges**

9  
10 [C#] public DataTable GetChanges();

11 [C++] public: DataTable\* GetChanges();

12 [VB] Public Function GetChanges() As DataTable

13 [JScript] public function GetChanges() : DataTable; Gets a copy of the  
14 **System.Data.DataTable** containing all changes made to it since it was last  
15 loaded, or since **System.Data.DataTable.AcceptChanges** was called.

16  
17 *Description*

18 Gets a copy of the **System.Data.DataTable** that contains all changes made  
19 to it since it was loaded or **System.Data.DataTable.AcceptChanges** was last  
20 called.

21 *Return Value:* A copy of the changes from this **System.Data.DataTable** that can  
22 have actions performed on it and subsequently be merged back in using  
23 **System.Data.DataSet.Merge(System.Data.DataSet)** , or **null** if none are found.

24 Gets a copy of the **System.Data.DataTable** that contains all changes made  
25 to it since it was loaded or **System.Data.DataTable.AcceptChanges** was last

called. This copy is particularly designed so that it can be merged back in to this original **System.Data.DataTable** . Relationship constraints may cause Unchanged parent rows to be included. If no rows of these rowStates are found, this method returns **null** .

#### GetChanges

[C#] public DataTable GetChanges(DataRowState rowStates);

[C++] public: DataTable\* GetChanges(DataRowState rowStates);

[VB] Public Function GetChanges(ByVal rowStates As DataRowState) As  
DataTable

[JScript] public function GetChanges(rowStates : DataRowState) : DataTable;

#### Description

Gets a copy of the **System.Data.DataTable** containing all changes made to it since it was last loaded, or since **System.Data.DataTable.AcceptChanges** was called, filtered by **System.Data.DataRowState** .

*Return Value:* A filtered copy of the **System.Data.DataTable** that can have actions performed on it, and subsequently be merged back in using **System.Data.DataSet.Merge(System.Data.DataSet)** . If no rows of the desired **System.Data.DataRowState** are found, the method returns **null** .

The **System.Data.DataTable.GetChanges** method is used to produce a second **System.Data.DataTable** object which contains only the changes introduced into the original. Use the *rowStates* argument to specify the type of changes the new object should include. One of the **System.Data.DataRowState** values.

## GetErrors

```
[C#] public DataRow[] GetErrors();  
[C++] public: DataRow* GetErrors() [];  
[VB] Public Function GetErrors() As DataRow()  
[JScript] public function GetErrors() : DataRow[];
```

### *Description*

Gets an array of **System.Data.DataRow** objects that contain errors.

*Return Value:* An array of **System.Data.DataRow** objects that have errors.

Invoke **System.Data.DataTable.GetErrors** after invoking the **System.Data.DataSet** class's **System.Data.DataSet.GetChanges** method. Also, be sure you don't invoke the **System.Data.DataTable.AcceptChanges** on the **System.Data.DataTable** until after all errors have been successfully resolved, and the **System.Data.DataSet** re-submitted for updating.

## GetRowType

```
[C#] protected virtual Type GetRowType();  
[C++] protected: virtual Type* GetRowType();  
[VB] Overridable Protected Function GetRowType() As Type  
[JScript] protected function GetRowType() : Type;
```

### *Description*

Gets the row type.

*Return Value:* The **System.Type** of the row.

## ImportRow

```
[C#] public void ImportRow(DataRow row);  
[C++] public: void ImportRow(DataRow* row);  
[VB] Public Sub ImportRow(ByVal row As DataRow)  
[JScript] public function ImportRow(row : DataRow);
```

### *Description*

Copies a **System.Data.DataRow** , including original and current values, **System.Data.DataRowState** values, and errors, into a **System.Data.DataTable** .

A **System.Data.DataRow**, including original and current values, **System.Data.DataRowState** values, and errors.

## LoadDataRow

```
[C#] public DataRow LoadDataRow(object[] values, bool fAcceptChanges);  
[C++] public: DataRow* LoadDataRow(Object* values __gc[], bool  
fAcceptChanges);  
[VB] Public Function LoadDataRow(ByVal values() As Object, ByVal  
fAcceptChanges As Boolean) As DataRow  
[JScript] public function LoadDataRow(values : Object[], fAcceptChanges :  
Boolean) : DataRow;
```

### *Description*

Finds and updates a specific row. If no matching row is found, a new row is created using the given values.

*Return Value:* The new **System.Data.DataRow** .

The **System.Data.DataTable.LoadDataRow(System.Object[],System.Boolean)** method takes an array of values and finds the matching value(s) in the primary key column(s). An array of values used to create the new row. **true** to accept changes; otherwise, **false**.

**DataRow**

[C#] public DataRow DataRow();  
[C++] public: DataRow\* DataRow();  
[VB] Public Function DataRow() As DataRow  
[JScript] public function DataRow() : DataRow;

### *Description*

Creates a new **System.Data.DataRow** with the same schema as the table.

*Return Value:* A **System.Data.DataRow** with the same schema as the **System.Data.DataTable** .

You must use the **System.Data.DataTable.NewRow** method to create new **System.Data.DataRow** objects with the same schema as the **System.Data.DataTable** . After creating a **System.Data.DataRow** , you can add it to the **System.Data.DataRowCollection** , through the **System.Data.DataTable** object's **System.Data.DataTable.Rows** property.

**DataRowArray**

1  
2 [C#] protected internal DataRow[] NewRowArray(int size);  
3 [C++] protected public: DataRow\* NewRowArray(int size) [];  
4 [VB] Protected Friend Dim Function NewRowArray(ByVal size As Integer) As  
5 DataRow()  
6 [JScript] package function NewRowArray(size : int) : DataRow[];

7  
8 *Description*

9  
10 NewRowFromBuilder

11  
12 [C#] protected virtual DataRow NewRowFromBuilder(DataRowBuilder builder);  
13 [C++] protected: virtual DataRow\* NewRowFromBuilder(DataRowBuilder\*  
14 builder);  
15 [VB] Overridable Protected Function NewRowFromBuilder(ByVal builder As  
16 DataRowBuilder) As DataRow  
17 [JScript] protected function NewRowFromBuilder(builder : DataRowBuilder) :  
18 DataRow;

19  
20 *Description*

21 This is what a subclassed dataSet overrides to create a new row.

22 OnColumnChanged

23  
24 [C#] protected virtual void OnColumnChanged(DataColumnChangeEventArgs e);  
25 [C++] protected: virtual void OnColumnChanged(DataColumnChangeEventArgs\*

e);

[VB] Overridable Protected Sub OnColumnChanged(ByVal e As

DataColumnChangeEventArgs)

[JScript] protected function OnColumnChanged(e :

DataColumnChangeEventArgs);

### *Description*

Raises the **System.Data.DataTable.ColumnChanged** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Data.DataColumnChangeEventArgs** that contains the event data.

### **OnColumnChanging**

[C#] protected virtual void OnColumnChanging(DataColumnChangeEventArgs

e);

[C++] protected: virtual void

OnColumnChanging(DataColumnChangeEventArgs\* e);

[VB] Overridable Protected Sub OnColumnChanging(ByVal e As

DataColumnChangeEventArgs)

[JScript] protected function OnColumnChanging(e :

DataColumnChangeEventArgs);

### *Description*

Raises the **System.Data.DataTable.ColumnChanging** event.

1 Raising an event invokes the event handler through a delegate. For an  
2 overview, see . A **System.Data.DataColumnChangeEventArgs** that contains the  
3 event data.

#### 4 OnPropertyChanging

5  
6 [C#] protected internal virtual void

7 OnPropertyChanging(PropertyChangedEventArgs pcevent);

8 [C++] protected public: virtual void

9 OnPropertyChanging(PropertyChangedEventArgs\* pcevent);

10 [VB] Overridable Protected Friend Dim Sub OnPropertyChanging(ByVal pcevent

11 As PropertyChangedEventArgs)

12 [JScript] package function OnPropertyChanging(pcevent :

13 PropertyChangedEventArgs);

#### 14 15 *Description*

16 Raises the

17 **System.Data.DataTable.OnPropertyChanging(System.ComponentModel.Pro**  
18 **pertyChangedEventArgs)** event.

19 Raising an event invokes the event handler through a delegate. For an  
20 overview, see . A **System.ComponentModel.PropertyChangedEventArgs** that  
21 contains the event data.

#### 22 OnRemoveColumn

23  
24 [C#] protected internal virtual void OnRemoveColumn(DataColumn column);

25 [C++] protected public: virtual void OnRemoveColumn(DataColumn\* column);

[VB] Overridable Protected Friend Dim Sub OnRemoveColumn(ByVal column  
As DataColumn)

[JScript] package function OnRemoveColumn(column : DataColumn);

#### *Description*

Notifies the **System.Data.DataTable** that a **System.Data.DataColumn** is  
being removed.

Raising an event invokes the event handler through a delegate. For more  
information, see . The **System.Data.DataColumn** being removed.

#### **OnRowChanged**

[C#] protected virtual void OnRowChanged(DataRowChangeEventArgs e);

[C++] protected: virtual void OnRowChanged(DataRowChangeEventArgs\* e);

[VB] Overridable Protected Sub OnRowChanged(ByVal e As  
DataRowChangeEventArgs)

[JScript] protected function OnRowChanged(e : DataRowChangeEventArgs);

#### *Description*

Raises the **System.Data.DataTable.RowChanged** event.

Raising an event invokes the event handler through a delegate. For an  
overview, see . A **System.Data.DataRowChangeEventArgs** that contains the  
event data.

#### **OnRowChanging**

[C#] protected virtual void OnRowChanging(DataRowChangeEventArgs e);

1 [C++] protected: virtual void OnRowChanging(DataRowChangeEventArgs\* e);

2 [VB] Overridable Protected Sub OnRowChanging(ByVal e As

3 DataRowChangeEventArgs)

4 [JScript] protected function OnRowChanging(e : DataRowChangeEventArgs);

5  
6 *Description*

7       Raises the **System.Data.DataTable.RowChanging** event.

8       Raising an event invokes the event handler through a delegate. For an  
9 overview, see . A **System.Data.DataRowChangeEventArgs** that contains the  
10 event data.

11       **OnRowDeleted**

12  
13 [C#] protected virtual void OnRowDeleted(DataRowChangeEventArgs e);

14 [C++] protected: virtual void OnRowDeleted(DataRowChangeEventArgs\* e);

15 [VB] Overridable Protected Sub OnRowDeleted(ByVal e As

16 DataRowChangeEventArgs)

17 [JScript] protected function OnRowDeleted(e : DataRowChangeEventArgs);

18  
19 *Description*

20       Raises the **System.Data.DataTable.RowDeleted** event.

21       Raising an event invokes the event handler through a delegate. For an  
22 overview, see . A **System.Data.DataRowChangeEventArgs** that contains the  
23 event data.

24       **OnRowDeleting**

[C#] protected virtual void OnRowDeleting(DataRowChangeEventArgs e);  
 [C++] protected: virtual void OnRowDeleting(DataRowChangeEventArgs\* e);  
 [VB] Overridable Protected Sub OnRowDeleting(ByVal e As  
 DataRowChangeEventArgs)  
 [JScript] protected function OnRowDeleting(e : DataRowChangeEventArgs);

### *Description*

Raises the  
**System.Data.DataTable.OnRowDeleting(System.Data.DataRowChangeEvent  
 Args)** event.

Raising an event invokes the event handler through a delegate. For an  
 overview, see . A **System.Data.DataRowChangeEventArgs** that contains the  
 event data.

### **RejectChanges**

[C#] public void RejectChanges();  
 [C++] public: void RejectChanges();  
 [VB] Public Sub RejectChanges()  
 [JScript] public function RejectChanges();

### *Description*

Rolls back all changes that have been made to the table since it was loaded,  
 or the last time **System.Data.DataTable.AcceptChanges** was called.

When **System.Data.DataTable.RejectChanges** is called, any **System.Data.DataRow** objects that are still in edit-mode cancel their edits. New rows are removed. Rows with the **System.Data.DataRowState** set to **Modified** or **Deleted** return back to their original state.

#### Reset

[C#] public virtual void Reset();  
[C++] public: virtual void Reset();  
[VB] Overridable Public Sub Reset()  
[JScript] public function Reset();

#### *Description*

Resets the **System.Data.DataTable** to its original state.

#### Select

[C#] public DataRow[] Select();  
[C++] public: DataRow\* Select() [];  
[VB] Public Function Select() As DataRow()  
[JScript] public function Select() : DataRow[]; Gets an array of **System.Data.DataRow** objects.

#### *Description*

Gets an array of all **System.Data.DataRow** objects.

*Return Value:* An array of **System.Data.DataRow** objects.

The method returns the current rows in order of primary key (or lacking one, order of addition.) The following example returns an array of **System.Data.DataRow** objects through the **System.Data.DataTable.Select** method.

Select

```
[C#] public DataRow[] Select(string filterExpression);  
[C++] public: DataRow* Select(String* filterExpression) [];  
[VB] Public Function Select(ByVal filterExpression As String) As DataRow()  
[JScript] public function Select(filterExpression : String) : DataRow[];
```

#### *Description*

Gets an array of all **System.Data.DataRow** objects that match the filter criteria in order of primary key (or lacking one, order of addition.)

*Return Value:* An array of **System.Data.DataRow** objects.

To create the *filterExpression* argument, use the same rules that apply to the **System.Data.DataColumn** class's **System.Data.DataColumn.Expression** property value for creating filters. The criteria to use to filter the rows.

Select

```
[C#] public DataRow[] Select(string filterExpression, string sort);  
[C++] public: DataRow* Select(String* filterExpression, String* sort) [];  
[VB] Public Function Select(ByVal filterExpression As String, ByVal sort As  
String) As DataRow()  
[JScript] public function Select(filterExpression : String, sort : String) :
```

1 DataRow[];

2  
3 *Description*

4 Gets an array of all **System.Data.DataRow** objects that match the filter  
5 criteria, in the the specified sort order.

6 *Return Value:* An array of **System.Data.DataRow** objects matching the filter  
7 expression.

8 To form the *filterExpression* argument, use the same rules for creating the  
9 **System.Data.DataColumn** class's **System.Data.DataColumn.Expression**  
10 property value. The *Sort* argument also uses the same rules for creating class's  
11 **System.Data.DataColumn.Expression** strings. The criteria to use to filter the  
12 rows. A string specifying the column and sort direction.

13 **Select**

14  
15 [C#] public DataRow[] Select(string filterExpression, string sort,  
16 DataRowState recordStates);

17 [C++] public: DataRow\* Select(String\* filterExpression, String\* sort,  
18 DataRowState recordStates) [];

19 [VB] Public Function Select(ByVal filterExpression As String, ByVal sort As  
20 String, ByVal recordStates As DataRowState) As DataRow()

21 [JScript] public function Select(filterExpression : String, sort : String, recordStates  
22 : DataRowState) : DataRow[];

23  
24 *Description*

25

Gets an array of all **System.Data.DataRow** objects that match the filter in the order of the sort, that match the specified state.

*Return Value:* An array of **System.Data.DataRow** objects.

To form the *filterExpression* argument, use the same rules for creating the **System.Data.DataColumn** class's **System.Data.DataColumn.Expression** property value. The *Sort* argument also uses the same rules for creating class's **System.Data.DataColumn.Expression** strings. The criteria to use to filter the rows. A string specifying the column and sort direction. One of the **System.Data.DataViewRowState** values.

**IListSource.GetList**

[C#] **IList IListSource.GetList();**

[C++] **IList\* IListSource::GetList();**

[VB] **Function GetList() As IList Implements IListSource.GetList**

[JScript] **function IListSource.GetList() : IList;**

**ISerializable.GetObjectData**

[C#] **void ISerializable.GetObjectData(SerializationInfo info, StreamingContext context);**

[C++] **void ISerializable::GetObjectData(SerializationInfo\* info, StreamingContext context);**

[VB] **Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext) Implements ISerializable.GetObjectData**

[JScript] **function ISerializable.GetObjectData(info : SerializationInfo, context : StreamingContext);**

## ToString

[C#] public override string ToString();

[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

### *Description*

Gets the **System.Data.DataTable.TableName** and **System.Data.DataTable.DisplayExpression**, if there is one as a concatenated string.

*Return Value:* A string consisting of the **System.Data.DataTable.TableName** and the **System.Data.DataTable.DisplayExpression** values.

Gets the **System.Data.DataTable.TableName** and **System.Data.DataTable.DisplayExpression** for the **System.Data.DataTable**.

DataTableCollection class (System.Data)

## ToString

### *Description*

Represents the collection of tables for the **System.Data.DataSet**.

The **System.Data.DataTableCollection** contains all of the **System.Data.DataTable** objects for a **System.Data.DataSet**. To access the **System.Data.DataTableCollection** of a **System.Data.DataSet**, use the **System.Data.DataSet.Tables** property.

1 Count  
2 IsReadOnly  
3 IsSynchronized  
4 Item  
5 ToString  
6 **System.Data.DataTable**

7  
8 *Description*

9 Gets the **System.Data.DataTable** specified by its index.  
10 The **System.Data.DataTableCollection.Contains(System.String)** method  
11 can be used to determine if a table with a specified index exists. The zero-based  
12 index of the **System.Data.DataTable** to find.

13 Item  
14 ToString

15  
16 [C#] public DataTable this[string name] {get;}  
17 [C++] public: \_\_property DataTable\* get\_Item(String\* name);  
18 [VB] Public Default ReadOnly Property Item(ByVal name As String) As  
19 DataTable  
20 [JScript] returnValue = DataTableCollectionObject.Item(name);

21  
22 *Description*

23 Gets the **System.Data.DataTable** in the collection with the given name  
24 (not case-sensitive).  
25

The **System.Data.DataTableCollection.Contains(System.String)** method can be used to determine if a table with a specified name or index exists. The name of the table to find.

List

ToString

[C#] protected override ArrayList List {get;}

[C++] protected: \_\_property virtual ArrayList\* get\_List();

[VB] Overrides Protected ReadOnly Property List As ArrayList

[JScript] protected function get List() : ArrayList;

#### *Description*

Gets the tables in the collection as an object.

SyncRoot

ToString

#### *Description*

Occurs when the collection is changed.

ToString

[C#] public event CollectionChangeEventHandler CollectionChanging;

[C++] public: \_\_event CollectionChangeEventHandler\* CollectionChanging;

[VB] Public Event CollectionChanging As CollectionChangeEventHandler

1  
2 *Description*

3 Occurs when the collection is changing.

4 To abort the change, the user should throw an exception in a  
5 **System.Data.DataColumnChangeEventHandler** event handler, and then catch  
6 the exception.

7 *Add*

8  
9 [C#] public virtual DataTable Add();

10 [C++] public: virtual DataTable\* Add();

11 [VB] Overridable Public Function Add() As DataTable

12 [JScript] public function Add() : DataTable;

13  
14 *Description*

15 Creates a new table with a default name and adds it to the collection.

16 *Return Value:* The newly created **System.Data.DataTable** .

17 Because no name is specified, the table is created with a default name,  
18 relative to its order of addition. The default name is "Table" where *i* = a new 1-  
19 based index.

20 *Add*

21  
22 [C#] public virtual void Add(DataTable table);

23 [C++] public: virtual void Add(DataTable\* table);

24 [VB] Overridable Public Sub Add(ByVal table As DataTable)

25 [JScript] public function Add(table : DataTable); Adds a **System.Data.DataTable**

1 to the collection.

2  
3 *Description*

4 *Adds the specified **System.Data.DataTable** to the collection.*

5 *The*

6 ***System.Data.DataTableCollection.OnCollectionChanged(System.ComponentModel***  
7 ***del.CollectionChangeEventArgs)** event occurs when a table is successfully added.*

8 ***System.Data.DataTable** to add.*

9 *Add*

10  
11 *[C#] public virtual DataTable Add(string name);*

12 *[C++] public: virtual DataTable\* Add(String\* name);*

13 *[VB] Overridable Public Function Add(ByVal name As String) As DataTable*

14 *[JScript] public function Add(name : String) : DataTable;*

15  
16 *Description*

17 *Creates a table with the given name and adds it to the collection.*

18 *Return Value: The newly created **System.Data.DataTable** .*

19 *If either a **null** or an empty string ("")* is passed in, a default name is given  
20 *to the newly created **System.Data.DataTable** . The name to give the created*  
21 ***System.Data.DataTable**.*

22 *AddRange*

23  
24 *[C#] public void AddRange(DataTable[] tables);*

25 *[C++] public: void AddRange(DataTable\* tables[]);*

1 *[VB] Public Sub AddRange(ByVal tables() As DataTable)*  
2 *[JScript] public function AddRange(tables : DataTable[]);*

3  
4 *Description*

5 *Copies the elements of the specified **System.Data.DataTable** array to the*  
6 *end of the collection. The array of **System.Data.DataTable** objects to add to the*  
7 *collection.*

8 *CanRemove*

9  
10 *[C#] public bool CanRemove(DataTable table);*  
11 *[C++] public: bool CanRemove(DataTable\* table);*  
12 *[VB] Public Function CanRemove(ByVal table As DataTable) As Boolean*  
13 *[JScript] public function CanRemove(table : DataTable) : Boolean;*

14  
15 *Description*

16 *Verifies if the specified **System.Data.DataTable** can be removed from the*  
17 *collection.*

18 *Return Value: **true** if the table can be removed; otherwise, **false**. A*

19 ***System.Data.DataTable** in the collection.*

20 *Clear*

21  
22 *[C#] public void Clear();*  
23 *[C++] public: void Clear();*  
24 *[VB] Public Sub Clear()*  
25 *[JScript] public function Clear();*

## *Description*

*Clears the collection of any tables.*

## *Contains*

*[C#] public bool Contains(string name);*

*[C++] public: bool Contains(String\* name);*

*[VB] Public Function Contains(ByVal name As String) As Boolean*

*[JScript] public function Contains(name : String) : Boolean;*

## *Description*

*Checks if a table, specified by name, exists in the collection.*

*Return Value: **true** if the specified table exists; otherwise, **false** .*

*The **System.Data.DataTable** object's name is specified by the **System.Data.DataTable.TableName** property. If you add a **System.Data.DataTable** to the **System.Data.DataTableCollection** with the **System.Data.DataTableCollection.Add(System.Data.DataTable)** method, passing no arguments, the table is given a default name such as Table1, Table2, and so on. The table name to check for.*

## *IndexOf*

*[C#] public virtual int IndexOf(DataTable table);*

*[C++] public: virtual int IndexOf(DataTable\* table);*

*[VB] Overridable Public Function IndexOf(ByVal table As DataTable) As Integer*

*[JScript] public function IndexOf(table : DataTable) : int; Gets the index of a*

specified table.

### Description

*Gets the index of a specified **System.Data.DataTable** .*

*Return Value: The 0-based index of the table, or -1 if the table isn't found in the collection.*

*Use the*

***System.Data.DataTableCollection.IndexOf(System.Data.DataTable)** method when it's necessary to know the exact index of a given table. The **System.Data.DataTable** to search for.*

### *IndexOf*

*[C#] public virtual int IndexOf(string tableName);*

*[C++] public: virtual int IndexOf(String\* tableName);*

*[VB] Overridable Public Function IndexOf(ByVal tableName As String) As*

*Integer*

*[JScript] public function IndexOf(tableName : String) : int;*

### Description

*Gets the index of the table with the given name (case insensitive), or -1 if the table doesn't exist in the collection.*

*Return Value: The index of the table with the name, or -1 if the table doesn't exist in the collection.*

*The name of a **System.Data.DataTable** is set with the **System.Data.DataTable.TableName** property. The name to look for.*

## *OnCollectionChanged*

*[C#] protected virtual void OnCollectionChanged(CollectionChangeEventArgs ccevent);*

*[C++] protected: virtual void*

*OnCollectionChanged(CollectionChangeEventArgs\* ccevent);*

*[VB] Overridable Protected Sub OnCollectionChanged(ByVal ccevent As CollectionChangeEventArgs)*

*[JScript] protected function OnCollectionChanged(ccevent : CollectionChangeEventArgs);*

## *Description*

*Raises the*

***System.Data.DataTableCollection.OnCollectionChanged(System.ComponentModel.CollectionChangeEventArgs)** event.*

*Raising an event invokes the event handler through a delegate. For an overview, see . A **System.ComponentModel.CollectionChangeEventArgs** that contains the event data.*

## *OnCollectionChanging*

*[C#] protected internal virtual void*

*OnCollectionChanging(CollectionChangeEventArgs ccevent);*

*[C++] protected public: virtual void*

*OnCollectionChanging(CollectionChangeEventArgs\* ccevent);*

*[VB] Overridable Protected Friend Dim Sub OnCollectionChanging(ByVal*

1 *ccevent As CollectionChangeEventArgs)*

2 *[JScript] package function OnCollectionChanging(ccevent :*  
3 *CollectionChangeEventArgs);*

4  
5 *Description*

6 *Raises the*  
7 ***System.Data.DataTableCollection.OnCollectionChanging(System.ComponentM***  
8 ***odel.CollectionChangeEventArgs) event.***

9 *Raising an event invokes the event handler through a delegate. For an*  
10 *overview, see . A **System.ComponentModel.CollectionChangeEventArgs** that*

11 *contains the event data.*

12 *Remove*  
13  
14 *[C#] public void Remove(DataTable table);*

15 *[C++] public: void Remove(DataTable\* table);*

16 *[VB] Public Sub Remove(ByVal table As DataTable)*

17 *[JScript] public function Remove(table : DataTable); Removes a table from the*  
18 *collection.*

19  
20 *Description*

21 *Removes the specified table from the collection.*

22 *The*  
23 ***System.Data.DataTableCollection.OnCollectionChanged(System.ComponentMo***  
24 ***del.CollectionChangeEventArgs) event occurs when a table is succesfully***  
25 ***removed. The **System.Data.DataTable** to remove.***

## *Remove*

```
[C#] public void Remove(string name);  
[C++] public: void Remove(String* name);  
[VB] Public Sub Remove(ByVal name As String)  
[JScript] public function Remove(name : String);
```

### *Description*

*Removes the table with a specified name from the collection.*

*The*

***System.Data.DataTableCollection.OnCollectionChanged(System.ComponentModel.CollectionChangeEventArgs)*** event occurs when a table is successfully removed. The name of the ***System.Data.DataTable*** to remove.

## *RemoveAt*

```
[C#] public void RemoveAt(int index);  
[C++] public: void RemoveAt(int index);  
[VB] Public Sub RemoveAt(ByVal index As Integer)  
[JScript] public function RemoveAt(index : int);
```

### *Description*

*Removes the table at the given index from the collection. The collection doesn't have a table at this index.*

*The*

***System.Data.DataTableCollection.OnCollectionChanged(System.ComponentModelMo***

*del.CollectionChangeEventArgs) event occurs when a table is successfully removed. The index at which to remove a table.*

*DataGridView class (System.Data)*

*ToString*

### *Description*

*Represents a databindable, customized view of a **System.Data.DataTable** for sorting, filtering, searching, editing, and navigation.*

*A major function of the **System.Data.DataView** is to allow data binding on both Windows Forms and Web Forms.*

*DataGridView*

*Example Syntax:*

*ToString*

*[C#] public DataGridView();*

*[C++] public: DataGridView();*

*[VB] Public Sub New()*

*[JScript] public function DataGridView(); Initializes a new instance of the **System.Data.DataView** class.*

### *Description*

*Initializes a new instance of the **System.Data.DataView** class.*

*DataGridView*

*Example Syntax:*

## *ToString*

```
[C#] public DataView(DataTable table);  
[C++] public: DataView(DataTable* table);  
[VB] Public Sub New(ByVal table As DataTable)  
[JScript] public function DataView(table : DataTable);
```

## *Description*

*Initializes a new instance of the **System.Data.DataView** class with the specified **System.Data.DataTable** . A **System.Data.DataTable** to add to the **System.Data.DataView**.*

*DataView*

*Example Syntax:*

## *ToString*

```
[C#] public DataView(DataTable table, string RowFilter, string Sort,  
DataViewRowState RowState);  
[C++] public: DataView(DataTable* table, String* RowFilter, String* Sort,  
DataViewRowState RowState);  
[VB] Public Sub New(ByVal table As DataTable, ByVal RowFilter As String,  
ByVal Sort As String, ByVal RowState As DataViewRowState)  
[JScript] public function DataView(table : DataTable, RowFilter : String, Sort :  
String, RowState : DataViewRowState); Initializes a new instance of the  
System.Data.DataView class with the specified System.Data.DataTable .
```

*AllowDelete*

## *ToString*

[C#] *public bool AllowDelete {get; set;}*

[C++] *public: \_\_property bool get\_AallowDelete();public: \_\_property void  
set\_AallowDelete(bool);*

[VB] *Public Property AllowDelete As Boolean*

[JScript] *public function get AllowDelete() : Boolean;public function set  
AllowDelete(Boolean);*

### *Description*

*Sets or gets a value indicating whether deletes are allowed.*

*AllowEdit*

*ToString*

[C#] *public bool AllowEdit {get; set;}*

[C++] *public: \_\_property bool get\_AallowEdit();public: \_\_property void  
set\_AallowEdit(bool);*

[VB] *Public Property AllowEdit As Boolean*

[JScript] *public function get AllowEdit() : Boolean;public function set  
AllowEdit(Boolean);*

### *Description*

*Gets or sets a value indicating whether edits are allowed.*

*AllowNew*

*ToString*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public bool AllowNew {get; set;}

[C++] public: __property bool get_AllowNew();public: __property void
set_AllowNew(bool);

[VB] Public Property AllowNew As Boolean

[JScript] public function get AllowNew() : Boolean;public function set
AllowNew(Boolean);
```

#### Description

*Gets or sets a value indicating whether the new rows can be added using the **System.Data.DataView.AddNew** method.*

*ApplyDefaultSort*

*ToString*

```
[C#] public bool ApplyDefaultSort {get; set;}

[C++] public: __property bool get_ApplyDefaultSort();public: __property void
set_ApplyDefaultSort(bool);

[VB] Public Property ApplyDefaultSort As Boolean

[JScript] public function get ApplyDefaultSort() : Boolean;public function set
ApplyDefaultSort(Boolean);
```

#### Description

*Gets or sets a value indicating whether to use the default sort.*

*Container*

*Count*

*ToString*

*Description*

*Gets the number of records in the **System.Data.DataView** after **System.Data.DataView.RowFilter** and **System.Data.DataView.RowStateFilter** have been applied.*

*DataViewManager*

*ToString*

*[C#] public DataViewManager DataViewManager {get;}*

*[C++] public: \_\_property DataViewManager\* get\_DataViewManager();*

*[VB] Public ReadOnly Property DataViewManager As DataViewManager*

*[JScript] public function get DataViewManager() : DataViewManager;*

*Description*

*Gets the **System.Data.DataView** associated with this view .*

*DesignMode*

*Events*

*IsOpen*

*ToString*

*Description*

1 Gets a value indicating whether the data source is currently open and  
2 projecting views of data on the **System.Data.DataTable** .

3 A **System.Data.DataView** is a "view" on a **System.Data.DataTable** because  
4 it provides custom sorting and filtering of the data. The  
5 **System.Data.DataView.IsOpen** property can be queried to determine if a  
6 **System.Data.DataView** has been opened using the **System.Data.DataView.Open**  
7 method.

8 Item

9 ToString

10  
11 [C#] public DataRowView this[int recordIndex] {get;}

12 [C++] public: \_\_property DataRowView\* get\_Item(int recordIndex);

13 [VB] Public Default ReadOnly Property Item(ByVal recordIndex As Integer) As  
14 DataRowView

15 [JScript] returnValue = DataViewObject.Item(recordIndex);

16  
17 Description

18 Gets a row of data from a specified table. The index of a record in the  
19 **System.Data.DataTable**.

20 RowFilter

21 ToString

22  
23 [C#] public virtual string RowFilter {get; set;}

24 [C++] public: \_\_property virtual String\* get\_RowFilter();public: \_\_property  
25 virtual void set\_RowFilter(String\*);

1 *[VB] Overridable Public Property RowFilter As String*

2 *[JScript] public function get RowFilter() : String;public function set*

3 *RowFilter(String);*

4  
5 *Description*

6 *Gets or sets the expression used to filter which rows are viewed in the*  
7 ***System.Data.DataView** .*

8 *To form a **System.Data.DataView.RowFilter** value, specify the name of a*  
9 *column followed by an operator and a value to filter on. The value must be in*  
10 *quotes. For example: "LastName = 'Smith'" See the **System.Data.DataColumn***  
11 *class's **System.Data.DataColumn.Expression** property for more information.*

12 *RowStateFilter*

13 *ToString*

14  
15 *[C#] public DataRowState RowStateFilter {get; set;}*

16 *[C++] public: \_\_property DataRowState get\_RowStateFilter();public:*

17 *\_\_property void set\_RowStateFilter(DataRowState);*

18 *[VB] Public Property RowStateFilter As DataRowState*

19 *[JScript] public function get RowStateFilter() : DataRowState;public*

20 *function set RowStateFilter(DataRowState);*

21  
22 *Description*

23 *Gets or sets the row state filter used in the **System.Data.DataView** .*

24 *Only rows that have been deleted using the*

25 ***System.Data.DataView.Delete(System.Int32)** method will have their*

***System.Data.DataView.RowStateFilter*** value set to ***Deleted*** . Those rows added using the ***System.Data.DataView.AddNew*** method will similarly have the property set to ***Added*** .

*Site*

*Sort*

*ToString*

*Description*

*Gets or sets the sort column or columns, and sort order for the table.*

*See the ***System.Data.DataColumn.Expression*** property of*

***System.Data.DataColumn*** for more details on forming a

***System.Data.DataView.Sort*** expression.

*Table*

*ToString*

***[C#] public DataTable Table {get; set;}***

***[C++] public: \_\_property DataTable\* get\_Table();public: \_\_property void***

***set\_Table(DataTable\*);***

***[VB] Public Property Table As DataTable***

***[JScript] public function get Table() : DataTable;public function set***

***Table(DataTable);***

*Description*

*Gets or sets the source ***System.Data.DataTable*** .*

The **System.Data.DataTable** also has a **System.Data.DataTable.DefaultView** property which returns the default **System.Data.DataView** for the table. For example, if you wish to create a custom view on the table, set the **System.Data.DataView.RowFilter** on the **System.Data.DataView** returned by the **System.Data.DataTable.DefaultView**.

*ToString*

*Description*

Occurs when the list managed by the **System.Data.DataView** changes.

*AddNew*

[C#] public virtual DataRowView AddNew();

[C++] public: virtual DataRowView\* AddNew();

[VB] Overridable Public Function AddNew() As DataRowView

[JScript] public function AddNew() : DataRowView;

*Description*

Adds a new row to the **System.Data.DataView**.

*Return Value:* A **System.Data.DataRowView**.

*BeginInit*

[C#] public void BeginInit();

[C++] public: \_\_sealed void BeginInit();

[VB] NotOverridable Public Sub BeginInit()

1 *[JScript] public function BeginInit();*

2  
3 *Description*

4 *Begins the initialization of a **System.Data.DataView** that is used on a form*  
5 *or used by another component. The initialization occurs at runtime.*

6 *The Visual Studio.NET design environment uses this method to start the*  
7 *initialization of a component that is used on a form or used by another component.*

8 *The **System.Data.DataView.EndInit** method ends the initialization. Using the*  
9 ***BeginInit** and **EndInit** methods prevents the control from being used before it is*  
10 *fully initialized.*

11 *Close*

12  
13 *[C#] protected void Close();*

14 *[C++] protected: void Close();*

15 *[VB] Protected Sub Close()*

16 *[JScript] protected function Close();*

17  
18 *Description*

19 *Closes the **System.Data.DataView** .*

20 *The method allows you to manually close the **System.Data.DataView** in*  
21 *derived classes. Use the corresponding **System.Data.DataView.Open** method to*  
22 *open the **System.Data.DataView** .*

23 *ColumnCollectionChanged*

24  
25 *[C#] protected virtual void ColumnCollectionChanged(object sender,*

*CollectionChangeEventArgs e);*

*[C++] protected: virtual void ColumnCollectionChanged(Object\* sender, CollectionChangeEventArgs\* e);*

*[VB] Overridable Protected Sub ColumnCollectionChanged(ByVal sender As Object, ByVal e As CollectionChangeEventArgs)*

*[JScript] protected function ColumnCollectionChanged(sender : Object, e : CollectionChangeEventArgs);*

### *Description*

*Occurs after a **System.Data.DataColumnCollection** has been changed successfully. The source of the event. A*

***System.ComponentModel.ListChangedEventArgs** that contains the event data.*

### *CopyTo*

*[C#] public void CopyTo(Array array, int index);*

*[C++] public: \_\_sealed void CopyTo(Array\* array, int index);*

*[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)*

*[JScript] public function CopyTo(array : Array, index : int);*

### *Description*

*Copies items into an array. Only for Web Forms Interfaces. array to copy into. index to start at.*

### *Delete*

1  
2 *[C#] public void Delete(int index);*  
3 *[C++] public: void Delete(int index);*  
4 *[VB] Public Sub Delete(ByVal index As Integer)*  
5 *[JScript] public function Delete(index : int);*

#### 6 7 *Description*

8 *Deletes a row at the specified index.*

9 *After deleting a **System.Data.DataRow** , its state changes to*  
10 ***DataViewRowState.Deleted** . You can roll back the deletion by calling*  
11 ***System.Data.DataTable.RejectChanges** on the **System.Data.DataTable** . The*  
12 *index of the row to delete.*

#### 13 *Dispose*

14  
15 *[C#] protected override void Dispose(bool disposing);*  
16 *[C++] protected: void Dispose(bool disposing);*  
17 *[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)*  
18 *[JScript] protected override function Dispose(disposing : Boolean);*

#### 19 20 *Description*

21 *Disposes of the resources (other than memory) used by the*  
22 ***System.Data.DataView** object.*

23 *Property change notifications between the **System.Data.DataView** and the*  
24 *underlying **System.Data.DataTable** stop after this method is called.*

#### 25 *EndInit*

1  
2 *[C#] public void EndInit();*

3 *[C++] public: \_\_sealed void EndInit();*

4 *[VB] NotOverridable Public Sub EndInit()*

5 *[JScript] public function EndInit();*

6  
7 *Description*

8 *Ends the initialization of a **System.Data.DataView** that is used on a form or*  
9 *used by another component. The initialization occurs at runtime.*

10 *The Visual Studio.NET design environment uses this method to end the*  
11 *initialization of a component that is used on a form or used by another component.*

12 *The **System.Data.DataView.BeginInit** method starts the initialization. Using the*  
13 ***BeginInit** and **EndInit** methods prevents the control from being used before it is*  
14 *fully initialized.*

15 *Find*

16  
17 *[C#] public int Find(object key);*

18 *[C++] public: int Find(Object\* key);*

19 *[VB] Public Function Find(ByVal key As Object) As Integer*

20 *[JScript] public function Find(key : Object) : int; Finds a row in the*  
21 ***System.Data.DataView** by the specified primary key value.*

22  
23 *Description*

24 *Finds a row in the **System.Data.DataView** by the specified primary key*  
25 *value.*

*Return Value:* The index of the row in the **System.Data.DataView** containing the primary key value specified; otherwise a null value if the primary key value does not exist. The object to search for.

#### *Find*

[C#] public int Find(object[] key);  
 [C++] public: int Find(Object\* key \_\_gc[]);  
 [VB] Public Function Find(ByVal key() As Object) As Integer  
 [JScript] public function Find(key : Object[]) : int;

#### *Description*

*Finds an array of rows in the System.Data.DataView by the specified primary key values.*

*Return Value:* The array of row indexes in the **System.Data.DataView** containing the primary key values specified; otherwise a null value if the primary key values do not exist. An array of values, typed as **System.Object**.

#### *FindRows*

[C#] public DataRowView[] FindRows(object key);  
 [C++] public: DataRowView\* FindRows(Object\* key) [];  
 [VB] Public Function FindRows(ByVal key As Object) As DataRowView()  
 [JScript] public function FindRows(key : Object) : DataRowView[]; Finds a row in the **System.Data.DataView** by the specified primary key value.

#### *FindRows*

1  
2 *[C#] public DataRowView[] FindRows(object[] key);*  
3 *[C++] public: DataRowView\* FindRows(Object\* key \_\_gc[]) [];*  
4 *[VB] Public Function FindRows(ByVal key() As Object) As DataRowView()*  
5 *[JScript] public function FindRows(key : Object[]) : DataRowView[]; Finds a*  
6 *row in the **System.Data.DataView** by the specified primary key values.*

#### *GetEnumerator*

8  
9 *[C#] public IEnumerator GetEnumerator();*  
10 *[C++] public: \_\_sealed IEnumerator\* GetEnumerator();*  
11 *[VB] NotOverridable Public Function GetEnumerator() As IEnumerator*  
12 *[JScript] public function GetEnumerator() : IEnumerator;*

#### *Description*

15 *Gets an enumerator for this **System.Data.DataView** .*  
16 *Return Value: An **System.Collections.IEnumerator** for navigating through the*  
17 *list.*

#### *IndexListChanged*

19  
20 *[C#] protected virtual void IndexListChanged(object sender,*  
21 *ListChangedEventArgs e);*  
22 *[C++] protected: virtual void IndexListChanged(Object\* sender,*  
23 *ListChangedEventArgs\* e);*  
24 *[VB] Overridable Protected Sub IndexListChanged(ByVal sender As Object,*  
25 *ByVal e As ListChangedEventArgs)*

*[JScript] protected function IndexListChanged(sender : Object, e : ListChangedEventArgs);*

#### *Description*

*Occurs after a **System.Data.DataView** has been changed successfully. The source of the event. A **System.ComponentModel.ListChangedEventArgs** that contains the event data.*

#### *OnListChanged*

*[C#] protected virtual void OnListChanged(ListChangedEventArgs e);*

*[C++] protected: virtual void OnListChanged(ListChangedEventArgs\* e);*

*[VB] Overridable Protected Sub OnListChanged(ByVal e As ListChangedEventArgs)*

*[JScript] protected function OnListChanged(e : ListChangedEventArgs);*

#### *Description*

*Raises the **System.Data.DataView.ListChanged** event. A **System.ComponentModel.ListChangedEventArgs** that contains the event data.*

#### *Open*

*[C#] protected void Open();*

*[C++] protected: void Open();*

*[VB] Protected Sub Open()*

*[JScript] protected function Open();*

## *Description*

*Opens a **System.Data.DataView** .*

*The method allows you to manually open the **System.Data.DataView** in derived classes. Use the corresponding **System.Data.DataView.Close** method to close the **System.Data.DataView** .*

## *Reset*

*[C#] protected void Reset();*

*[C++] protected: void Reset();*

*[VB] Protected Sub Reset()*

*[JScript] protected function Reset();*

## *Description*

*Reserved for internal use only.*

## *IList.Add*

*[C#] int IList.Add(object value);*

*[C++] int IList::Add(Object\* value);*

*[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add*

*[JScript] function IList.Add(value : Object) : int;*

## *IList.Clear*

*[C#] void IList.Clear();*

*[C++] void IList::Clear();*

```

1  [VB] Sub Clear() Implements IList.Clear
2  [JScript] function IList.Clear();
3      IList.Contains
4
5  [C#] bool IList.Contains(object value);
6  [C++] bool IList::Contains(Object* value);
7  [VB] Function Contains(ByVal value As Object) As Boolean Implements
8  IList.Contains
9  [JScript] function IList.Contains(value : Object) : Boolean;
10     IList.IndexOf
11
12  [C#] int IList.IndexOf(object value);
13  [C++] int IList::IndexOf(Object* value);
14  [VB] Function IndexOf(ByVal value As Object) As Integer Implements
15  IList.IndexOf
16  [JScript] function IList.IndexOf(value : Object) : int;
17     IList.Insert
18
19  [C#] void IList.Insert(int index, object value);
20  [C++] void IList::Insert(int index, Object* value);
21  [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
22  IList.Insert
23  [JScript] function IList.Insert(index : int, value : Object);
24     IList.Remove
25

```

```

1
2  [C#] void IList.Remove(object value);
3  [C++] void IList::Remove(Object* value);
4  [VB] Sub Remove(ByVal value As Object) Implements IList.Remove
5  [JScript] function IList.Remove(value : Object);
6      IList.RemoveAt
7
8  [C#] void IList.RemoveAt(int index);
9  [C++] void IList::RemoveAt(int index);
10 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt
11 [JScript] function IList.RemoveAt(index : int);
12     IBindingList.AddIndex
13
14 [C#] void IBindingList.AddIndex(PropertyDescriptor property);
15 [C++] void IBindingList::AddIndex(PropertyDescriptor* property);
16 [VB] Sub AddIndex(ByVal property As PropertyDescriptor) Implements
17     IBindingList.AddIndex
18 [JScript] function IBindingList.AddIndex(property : PropertyDescriptor);
19     IBindingList.AddNew
20
21 [C#] object IBindingList.AddNew();
22 [C++] Object* IBindingList::AddNew();
23 [VB] Function AddNew() As Object Implements IBindingList.AddNew
24 [JScript] function IBindingList.AddNew() : Object;
25     IBindingList.ApplySort

```

```

1
2  [C#] void IBindingList.ApplySort(PropertyDescriptor property, ListSortDirection
3  direction);
4  [C++] void IBindingList::ApplySort(PropertyDescriptor* property,
5  ListSortDirection direction);
6  [VB] Sub ApplySort(ByVal property As PropertyDescriptor, ByVal direction As
7  ListSortDirection) Implements IBindingList.ApplySort
8  [JScript] function IBindingList.ApplySort(property : PropertyDescriptor,
9  direction : ListSortDirection);
10         IBindingList.Find
11
12  [C#] int IBindingList.Find(PropertyDescriptor property, object key);
13  [C++] int IBindingList::Find(PropertyDescriptor* property, Object* key);
14  [VB] Function Find(ByVal property As PropertyDescriptor, ByVal key As Object)
15  As Integer Implements IBindingList.Find
16  [JScript] function IBindingList.Find(property : PropertyDescriptor, key : Object)
17  : int;
18         IBindingList.RemoveIndex
19
20  [C#] void IBindingList.RemoveIndex(PropertyDescriptor property);
21  [C++] void IBindingList::RemoveIndex(PropertyDescriptor* property);
22  [VB] Sub RemoveIndex(ByVal property As PropertyDescriptor) Implements
23  IBindingList.RemoveIndex
24  [JScript] function IBindingList.RemoveIndex(property : PropertyDescriptor);
25         IBindingList.RemoveSort

```

```

1
2  [C#] void IBindingList.RemoveSort();
3  [C++] void IBindingList::RemoveSort();
4  [VB] Sub RemoveSort() Implements IBindingList.RemoveSort
5  [JScript] function IBindingList.RemoveSort();
6      IList.GetItemProperties
7
8  [C#] PropertyDescriptorCollection
9  IList.GetItemProperties(PropertyDescriptor[] listAccessors);
10 [C++] PropertyDescriptorCollection*
11 IList::GetItemProperties(PropertyDescriptor* listAccessors[]);
12 [VB] Function GetItemProperties(ByVal listAccessors() As PropertyDescriptor)
13 As PropertyDescriptorCollection Implements IList.GetItemProperties
14 [JScript] function IList.GetItemProperties(listAccessors :
15 PropertyDescriptor[]) : PropertyDescriptorCollection;
16     IList.GetListName
17
18 [C#] string IList.GetListName(PropertyDescriptor[] listAccessors);
19 [C++] String* IList::GetListName(PropertyDescriptor* listAccessors[]);
20 [VB] Function GetListName(ByVal listAccessors() As PropertyDescriptor) As
21 String Implements IList.GetListName
22 [JScript] function IList.GetListName(listAccessors : PropertyDescriptor[]) :
23 String;
24     UpdateIndex
25

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*[C#] protected void UpdateIndex();*  
*[C++] protected: void UpdateIndex();*  
*[VB] Protected Sub UpdateIndex()*  
*[JScript] protected function UpdateIndex(); Reserved for internal use only.*

*Description*  
*Reserved for internal use only.*  
*UpdateIndex*

*[C#] protected virtual void UpdateIndex(bool force);*  
*[C++] protected: virtual void UpdateIndex(bool force);*  
*[VB] Overridable Protected Sub UpdateIndex(ByVal force As Boolean)*  
*[JScript] protected function UpdateIndex(force : Boolean);*

*Description*  
*Reserved for internal use only. Reserved for internal use only.*  
*DataManager class (System.Data)*  
*UpdateIndex*

*Description*  
*Contains a default System.Data.DataViewSettingCollection for each*  
*System.Data.DataTable in a System.Data.DataSet .*  
*DataManager*

*Example Syntax:*

*UpdateIndex*

*[C#] public DataViewManager();*

*[C++] public: DataViewManager();*

*[VB] Public Sub New()*

*[JScript] public function DataViewManager(); Initializes a new instance of the*

*System.Data.DataViewManager class.*

*Description*

*Initializes a new instance of the System.Data.DataViewManager class.*

*DataViewManager*

*Example Syntax:*

*UpdateIndex*

*[C#] public DataViewManager(DataSet dataSet);*

*[C++] public: DataViewManager(DataSet\* dataSet);*

*[VB] Public Sub New(ByVal dataSet As DataSet)*

*[JScript] public function DataViewManager(dataSet : DataSet);*

*Description*

*Initializes a new instance of the System.Data.DataViewManager class for the specified System.Data.DataSet . The name of the System.Data.DataSet to use.*

*Container*

*DataSet*

## *UpdateIndex*

### *Description*

*Gets or sets the name of the **System.Data.DataSet** to use with the **System.Data.DataViewManager** .*

*DataViewSettingCollectionString*

*UpdateIndex*

*[C#] public string DataViewSettingCollectionString {get; set;}*

*[C++] public: \_\_property String\* get\_DataViewSettingCollectionString();public:*

*\_\_property void set\_DataViewSettingCollectionString(String\*);*

*[VB] Public Property DataViewSettingCollectionString As String*

*[JScript] public function get DataViewSettingCollectionString() : String;public*

*function set DataViewSettingCollectionString(String);*

### *Description*

*Gets or sets a value used for code persistence.*

*A user should not call*

***System.Data.DataViewManager.DataViewSettingCollectionString** directly.*

*DataViewSettings*

*UpdateIndex*

*[C#] public DataViewSettingCollection DataViewSettings {get;}*

*[C++] public: \_\_property DataViewSettingCollection\* get\_DataViewSettings();*

1 *[VB] Public ReadOnly Property DataViewSettings As DataViewSettingCollection*

2 *[JScript] public function get DataViewSettings() : DataViewSettingCollection;*

3  
4 *Description*

5 *Gets the System.Data.DataViewSettingCollection for each*

6 *System.Data.DataTable in the System.Data.DataSet .*

7 *DesignMode*

8 *Events*

9 *Site*

10 *UpdateIndex*

11  
12  
13 *Description*

14 *Occurs a row is added to or deleted from a System.Data.DataView .*

15 *CreateDataView*

16  
17 *[C#] public DataView CreateDataView(DataTable table);*

18 *[C++] public: DataView\* CreateDataView(DataTable\* table);*

19 *[VB] Public Function CreateDataView(ByVal table As DataTable) As DataView*

20 *[JScript] public function CreateDataView(table : DataTable) : DataView;*

21  
22 *Description*

23 *Creates a System.Data.DataView for the specified System.Data.DataTable*

24 *. The name of the System.Data.DataTable to use in the System.Data.DataView.*

25 *OnListChanged*

[C#] protected virtual void OnListChanged(ListChangedEventArgs e);  
 [C++] protected: virtual void OnListChanged(ListChangedEventArgs\* e);  
 [VB] Overridable Protected Sub OnListChanged(ByVal e As  
 ListChangedEventArgs)  
 [JScript] protected function OnListChanged(e : ListChangedEventArgs);

#### Description

Raises the **System.Data.DataViewManager.ListChanged** event. A  
**System.ComponentModel.ListChangedEventArgs** that contains the event data.

#### RelationCollectionChanged

[C#] protected virtual void RelationCollectionChanged(object sender,  
 CollectionChangeEventArgs e);  
 [C++] protected: virtual void RelationCollectionChanged(Object\* sender,  
 CollectionChangeEventArgs\* e);  
 [VB] Overridable Protected Sub RelationCollectionChanged(ByVal sender As  
 Object, ByVal e As CollectionChangeEventArgs)  
 [JScript] protected function RelationCollectionChanged(sender : Object, e :  
 CollectionChangeEventArgs);

#### Description

Raises a **System.Data.DataRelationCollection.CollectionChanged** event  
 when a **System.Data.DataRelation** is added to or removed from the  
**System.Data.DataRelationCollection**. The source of the event. A

1 *System.ComponentModel.CollectionChangeEventArgs* that contains the event  
2 data.

3 *ICollection.CopyTo*

4  
5 [C#] void *ICollection.CopyTo(Array array, int index);*

6 [C++] void *ICollection::CopyTo(Array\* array, int index);*

7 [VB] Sub *CopyTo(ByVal array As Array, ByVal index As Integer) Implements*

8 *ICollection.CopyTo*

9 [JScript] function *ICollection.CopyTo(array : Array, index : int);*

10 *IEnumerable.GetEnumerator*

11  
12 [C#] *IEnumerator IEnumerable.GetEnumerator();*

13 [C++] *IEnumerator\* IEnumerable::GetEnumerator();*

14 [VB] Function *GetEnumerator() As IEnumerator Implements*

15 *IEnumerable.GetEnumerator*

16 [JScript] function *IEnumerable.GetEnumerator() : IEnumerator;*

17 *IList.Add*

18  
19 [C#] int *IList.Add(object value);*

20 [C++] int *IList::Add(Object\* value);*

21 [VB] Function *Add(ByVal value As Object) As Integer Implements IList.Add*

22 [JScript] function *IList.Add(value : Object) : int;*

23 *IList.Clear*

24  
25 [C#] void *IList.Clear();*

```

1  [C++] void IList::Clear();
2  [VB] Sub Clear() Implements IList.Clear
3  [JScript] function IList.Clear();
4      IList.Contains
5
6  [C#] bool IList.Contains(object value);
7  [C++] bool IList::Contains(Object* value);
8  [VB] Function Contains(ByVal value As Object) As Boolean Implements
9  IList.Contains
10 [JScript] function IList.Contains(value : Object) : Boolean;
11     IList.IndexOf
12
13 [C#] int IList.IndexOf(object value);
14 [C++] int IList::IndexOf(Object* value);
15 [VB] Function IndexOf(ByVal value As Object) As Integer Implements
16 IList.IndexOf
17 [JScript] function IList.IndexOf(value : Object) : int;
18     IList.Insert
19
20 [C#] void IList.Insert(int index, object value);
21 [C++] void IList::Insert(int index, Object* value);
22 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
23 IList.Insert
24 [JScript] function IList.Insert(index : int, value : Object);
25     IList.Remove

```

```

[C#] void IList.Remove(object value);
[C++] void IList::Remove(Object* value);
[VB] Sub Remove(ByVal value As Object) Implements IList.Remove
[JScript] function IList.Remove(value : Object);

    IList.RemoveAt

[C#] void IList.RemoveAt(int index);
[C++] void IList::RemoveAt(int index);
[VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt
[JScript] function IList.RemoveAt(index : int);

    IBindingList.AddIndex

[C#] void IBindingList.AddIndex(PropertyDescriptor property);
[C++] void IBindingList::AddIndex(PropertyDescriptor* property);
[VB] Sub AddIndex(ByVal property As PropertyDescriptor) Implements
IBindingList.AddIndex
[JScript] function IBindingList.AddIndex(property : PropertyDescriptor);

    IBindingList.AddNew

[C#] object IBindingList.AddNew();
[C++] Object* IBindingList::AddNew();
[VB] Function AddNew() As Object Implements IBindingList.AddNew
[JScript] function IBindingList.AddNew() : Object;

    IBindingList.ApplySort
    
```

```

1
2  [C#] void IBindingList.ApplySort(PropertyDescriptor property, ListSortDirection
3  direction);
4  [C++] void IBindingList::ApplySort(PropertyDescriptor* property,
5  ListSortDirection direction);
6  [VB] Sub ApplySort(ByVal property As PropertyDescriptor, ByVal direction As
7  ListSortDirection) Implements IBindingList.ApplySort
8  [JScript] function IBindingList.ApplySort(property : PropertyDescriptor,
9  direction : ListSortDirection);
10      IBindingList.Find
11
12  [C#] int IBindingList.Find(PropertyDescriptor property, object key);
13  [C++] int IBindingList::Find(PropertyDescriptor* property, Object* key);
14  [VB] Function Find(ByVal property As PropertyDescriptor, ByVal key As Object)
15  As Integer Implements IBindingList.Find
16  [JScript] function IBindingList.Find(property : PropertyDescriptor, key : Object)
17  : int;
18      IBindingList.RemoveIndex
19
20  [C#] void IBindingList.RemoveIndex(PropertyDescriptor property);
21  [C++] void IBindingList::RemoveIndex(PropertyDescriptor* property);
22  [VB] Sub RemoveIndex(ByVal property As PropertyDescriptor) Implements
23  IBindingList.RemoveIndex
24  [JScript] function IBindingList.RemoveIndex(property : PropertyDescriptor);
25      IBindingList.RemoveSort

```

```

1
2  [C#] void IBindingList.RemoveSort();
3  [C++] void IBindingList::RemoveSort();
4  [VB] Sub RemoveSort() Implements IBindingList.RemoveSort
5  [JScript] function IBindingList.RemoveSort();
6      IList.GetItemProperties
7
8  [C#] PropertyDescriptorCollection
9      IList.GetItemProperties(PropertyDescriptor[] listAccessors);
10 [C++] PropertyDescriptorCollection*
11     IList::GetItemProperties(PropertyDescriptor* listAccessors[]);
12 [VB] Function GetItemProperties(ByVal listAccessors() As PropertyDescriptor)
13     As PropertyDescriptorCollection Implements IList.GetItemProperties
14 [JScript] function IList.GetItemProperties(listAccessors :
15     PropertyDescriptor[]) : PropertyDescriptorCollection;
16     IList.GetListName
17
18 [C#] string IList.GetListName(PropertyDescriptor[] listAccessors);
19 [C++] String* IList::GetListName(PropertyDescriptor* listAccessors[]);
20 [VB] Function GetListName(ByVal listAccessors() As PropertyDescriptor) As
21     String Implements IList.GetListName
22 [JScript] function IList.GetListName(listAccessors : PropertyDescriptor[]) :
23     String;
24     TableCollectionChanged
25

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] protected virtual void TableCollectionChanged(object sender,
CollectionChangeEventArgs e);

[C++] protected: virtual void TableCollectionChanged(Object* sender,
CollectionChangeEventArgs* e);

[VB] Overridable Protected Sub TableCollectionChanged(ByVal sender As
Object, ByVal e As CollectionChangeEventArgs)

[JScript] protected function TableCollectionChanged(sender : Object, e :
CollectionChangeEventArgs);
```

*Description*

*Raises a **System.Data.DataTableCollection.CollectionChanged** event when a **System.Data.DataTable** is added to or removed from the **System.Data.DataTableCollection** . The source of the event. A **System.ComponentModel.CollectionChangeEventArgs** that contains the event data.*

*DataRowState enumeration (System.Data)*

*ToString*

*Description*

*Describes the version of data in a **System.Data.DataRow** .*

*The **System.Data.DataViewRowState** values are used either to retrieve a particular version of data from a **System.Data.DataRow** , or to determine what versions exist.*

1	<i>ToString</i>
2	
3	<i>[C#] public const DataRowState Added;</i>
4	<i>[C++] public: const DataRowState Added;</i>
5	<i>[VB] Public Const Added As DataRowState</i>
6	<i>[JScript] public var Added : DataRowState;</i>
7	
8	<i>Description</i>
9	<i>A new row.</i>
10	<i>ToString</i>
11	
12	<i>[C#] public const DataRowState CurrentRows;</i>
13	<i>[C++] public: const DataRowState CurrentRows;</i>
14	<i>[VB] Public Const CurrentRows As DataRowState</i>
15	<i>[JScript] public var CurrentRows : DataRowState;</i>
16	
17	<i>Description</i>
18	<i>Current rows including unchanged, new, and modified rows.</i>
19	<i>ToString</i>
20	
21	<i>[C#] public const DataRowState Deleted;</i>
22	<i>[C++] public: const DataRowState Deleted;</i>
23	<i>[VB] Public Const Deleted As DataRowState</i>
24	<i>[JScript] public var Deleted : DataRowState;</i>
25	

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

*A deleted row.*

*ToString*

*[C#] public const DataRowState ModifiedCurrent;*  
*[C++] public: const DataRowState ModifiedCurrent;*  
*[VB] Public Const ModifiedCurrent As DataRowState*  
*[JScript] public var ModifiedCurrent : DataRowState;*

*Description*

*A current version, which is a modified version of original data (see **ModifiedOriginal** ).*

*ToString*

*[C#] public const DataRowState ModifiedOriginal;*  
*[C++] public: const DataRowState ModifiedOriginal;*  
*[VB] Public Const ModifiedOriginal As DataRowState*  
*[JScript] public var ModifiedOriginal : DataRowState;*

*Description*

*The original version (although it has since been modified and is available as **ModifiedCurrent** ).*

*ToString*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const DataRowState None;
[C++] public: const DataRowState None;
[VB] Public Const None As DataRowState
[JScript] public var None : DataRowState;
```

*Description*

*None.*

*ToString*

```
[C#] public const DataRowState OriginalRows;
[C++] public: const DataRowState OriginalRows;
[VB] Public Const OriginalRows As DataRowState
[JScript] public var OriginalRows : DataRowState;
```

*Description*

*Original rows including unchanged and deleted rows.*

*ToString*

```
[C#] public const DataRowState Unchanged;
[C++] public: const DataRowState Unchanged;
[VB] Public Const Unchanged As DataRowState
[JScript] public var Unchanged : DataRowState;
```

*Description*

1        *An unchanged row.*  
 2        *DataRowSetting class (System.Data)*  
 3        *ToString*

6        *Description*

7                *Represents the default settings for ApplyDefaultSort, DataViewManager,*  
 8        *RowFilter, RowStateFilter, Sort, and Table for DataViews created from the*  
 9        *System.Data.DataViewManager .*

10        *ApplyDefaultSort*  
 11        *ToString*

13        *[C#] public bool ApplyDefaultSort {get; set;}*  
 14        *[C++] public: \_\_property bool get \_ApplyDefaultSort();public: \_\_property void*  
 15        *set \_ApplyDefaultSort(bool);*  
 16        *[VB] Public Property ApplyDefaultSort As Boolean*  
 17        *[JScript] public function get ApplyDefaultSort() : Boolean;public function set*  
 18        *ApplyDefaultSort(Boolean);*

20        *Description*

21                *Gets or sets a value indicating whether to use the default sort.*  
 22        *DataViewManager*  
 23        *ToString*

25        *[C#] public DataViewManager DataViewManager {get;}*

1 *[C++] public: \_\_property DataViewManager\* get\_DataViewManager();*  
 2 *[VB] Public ReadOnly Property DataViewManager As DataViewManager*  
 3 *[JScript] public function get DataViewManager() : DataViewManager;*

4  
 5 *Description*

6 *Gets the System.Data.DataViewManager that contains this*  
 7 *System.Data.DataViewSetting .*

8 *RowFilter*

9 *ToString*

10  
 11 *[C#] public string RowFilter {get; set;}*

12 *[C++] public: \_\_property String\* get\_RowFilter();public: \_\_property void*  
 13 *set\_RowFilter(String\*);*

14 *[VB] Public Property RowFilter As String*

15 *[JScript] public function get RowFilter() : String;public function set*  
 16 *RowFilter(String);*

17  
 18 *Description*

19 *Gets or sets the filter to apply in the System.Data.DataView .*

20 *RowStateFilter*

21 *ToString*

22  
 23 *[C#] public DataRowState RowStateFilter {get; set;}*

24 *[C++] public: \_\_property DataRowState get\_RowStateFilter();public:*  
 25 *\_\_property void set\_RowStateFilter(DataViewRowState);*

*[VB] Public Property RowStateFilter As DataRowState*

*[JScript] public function get RowStateFilter() : DataRowState;public*

*function set RowStateFilter(DataViewRowState);*

#### *Description*

*Gets or sets a value indicating whether to display Current, Deleted, Modified Current, ModifiedOriginal, New, Original, Unchanged, or no rows in the System.Data.DataView .*

*Sort*

*ToString*

*[C#] public string Sort {get; set;}*

*[C++] public: \_\_property String\* get \_Sort();public: \_\_property void*

*set \_Sort(String\*);*

*[VB] Public Property Sort As String*

*[JScript] public function get Sort() : String;public function set Sort(String);*

#### *Description*

*Gets or sets a value indicating the Sort to apply in the System.Data.DataView .*

*Table*

*ToString*

*[C#] public DataTable Table {get;}*

*[C++] public: \_\_property DataTable\* get \_Table();*

1 *[VB] Public ReadOnly Property Table As DataTable*

2 *[JScript] public function get Table() : DataTable;*

3

4 *Description*

5 *Gets the **System.Data.DataTable** to which the*

6 ***System.Data.DataViewSetting** properties apply.*

7 ***DataViewSettingCollection** class (**System.Data**)*

8 ***ToString***

9

10

11 *Description*

12 *Contains a read-only collection of **System.Data.DataViewSetting** objects*

13 *for each **System.Data.DataTable** in a **System.Data.DataSet** .*

14 *A user cannot add or remove a **DataViewSetting** from the collection, but*

15 *can change the properties of the **DataViewSetting** corresponding to a particular*

16 ***DataTable**. Adding or removing a **DataTable** from the **DataSet** adds or removes*

17 *the corresponding **DataViewSetting** from the collection.*

18 ***Count***

19 ***ToString***

20

21 *[C#] public virtual int Count {get;}*

22 *[C++] public: \_\_property virtual int get\_Count();*

23 *[VB] Overridable Public ReadOnly Property Count As Integer*

24 *[JScript] public function get Count() : int;*

25

## *Description*

*Gets the number of **System.Data.DataViewSetting** objects in the **System.Data.DataViewSettingCollection** .*

*The number of **System.Data.DataViewSetting** objects is the same as the number of **System.Data.DataTable** objects in the **System.Data.DataSet** .*

*IsReadOnly*

*ToString*

*[C#] public bool IsReadOnly {get;}*

*[C++] public: \_\_property bool get\_IsReadOnly();*

*[VB] Public ReadOnly Property IsReadOnly As Boolean*

*[JScript] public function get IsReadOnly() : Boolean;*

## *Description*

*Gets a value indicating whether the **System.Data.DataViewSettingCollection** is read-only.*

*IsSynchronized*

*ToString*

*[C#] public bool IsSynchronized {get;}*

*[C++] public: \_\_property bool get\_IsSynchronized();*

*[VB] Public ReadOnly Property IsSynchronized As Boolean*

*[JScript] public function get IsSynchronized() : Boolean;*

## Description

*Gets a value indicating whether access to the*

***System.Data.DataViewSettingCollection*** is synchronized (thread-safe).

*This property implements the **System.Collections.ICollection** interface.*

*Item*

*ToString*

*[C#] public virtual DataViewSetting this[DataTable table] {get; set;}*

*[C++] public: \_\_property virtual DataViewSetting\* get\_Item(DataTable\**

*table);public: \_\_property virtual void set\_Item(DataTable\* table,*

*DataViewSetting\*);*

*[VB] Overridable Public Default Property Item(ByVal table As DataTable) As*

*DataViewSetting*

*[JScript] returnValue =*

*DataViewSettingCollectionObject.Item(table);DataViewSettingCollectionObject.It*

*em(table) = returnValue; Gets the specified **System.Data.DataTable** from the*

*collection.*

## Description

*Gets the specified **System.Data.DataTable** object from the collection. The*

***System.Data.DataTable***to find.

*Item*

*ToString*

```

1
2  [C#] public virtual DataViewSetting this[string tableName] {get;}
3  [C++] public: __property virtual DataViewSetting* get_Item(String*
4  tableName);
5  [VB] Overridable Public Default ReadOnly Property Item(ByVal tableName As
6  String) As DataViewSetting
7  [JScript] returnValue = DataViewSettingCollectionObject.Item(tableName);
8
9  Description
10     Gets the specified System.Data.DataTable from the collection. The name of
11     the System.Data.DataTable to find.
12     Item
13     ToString
14
15  [C#] public virtual DataViewSetting this[int index] {get; set;}
16  [C++] public: __property virtual DataViewSetting* get_Item(int index);public:
17  __property virtual void set_Item(int index, DataViewSetting*);
18  [VB] Overridable Public Default Property Item(ByVal index As Integer) As
19  DataViewSetting
20  [JScript] returnValue =
21  DataViewSettingCollectionObject.Item(index);DataViewSettingCollectionObject.I
22  tem(index) = returnValue;
23
24  Description
25

```

*Gets the **System.Data.DataTable** specified by its index. The zero-based index of the **System.Data.DataTable** to find.*

*SyncRoot*

*ToString*

*[C#] public object SyncRoot {get;}*

*[C++] public: \_\_property Object\* get\_SyncRoot();*

*[VB] Public ReadOnly Property SyncRoot As Object*

*[JScript] public function get SyncRoot() : Object;*

#### *Description*

*Gets an object that can be used to synchronize access to the **System.Data.DataViewSettingCollection** .*

*This property implements the **System.Collections.ICollection** interface.*

*CopyTo*

*[C#] public void CopyTo(Array ar, int index);*

*[C++] public: \_\_sealed void CopyTo(Array\* ar, int index);*

*[VB] NotOverridable Public Sub CopyTo(ByVal ar As Array, ByVal index As Integer)*

*[JScript] public function CopyTo(ar : Array, index : int);*

#### *Description*

*Copies the elements of the **System.Data.DataViewSettingCollection** to the specified array. An **System.Array** to which to copy*

1 *System.Data.DataViewSettingCollection* elements. The starting index of the  
2 array.

3 *GetEnumerator*

4  
5 *[C#] public IEnumerator GetEnumerator();*

6 *[C++] public: \_\_sealed IEnumerator\* GetEnumerator();*

7 *[VB] NotOverridable Public Function GetEnumerator() As IEnumerator*

8 *[JScript] public function GetEnumerator() : IEnumerator;*

9  
10 *Description*

11 *Gets an IEnumerator for the collection.*

12 *DBConcurrencyException* class (System.Data)

13 *ToString*

14  
15  
16 *Description*

17 *The exception that is thrown by the DataAdapter during the update*  
18 *operation if the number of rows affected equals zero.*

19 *The DataAdapter examines the number of rows affected by the execution of*  
20 *each insert, update, or delete operation, and throws this exception if the number*  
21 *equals zero. This is usually the result of a concurrency violation.*

22 *DBConcurrencyException*

23 *Example Syntax:*

24 *ToString*

25

```

[C#] public DBConcurrencyException(string message);
[C++] public: DBConcurrencyException(String* message);
[VB] Public Sub New(ByVal message As String)
[JavaScript] public function DBConcurrencyException(message : String); Initializes
a new instance of the System.Data.DBConcurrencyException class.
    
```

#### *Description*

*Initializes a new instance of the **System.Data.DBConcurrencyException** class. The text string describing the details of the exception.*

*DBConcurrencyException*

*Example Syntax:*

*ToString*

```

[C#] public DBConcurrencyException(string message, Exception inner);
[C++] public: DBConcurrencyException(String* message, Exception* inner);
[VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
[JavaScript] public function DBConcurrencyException(message : String, inner :
Exception);
    
```

#### *Description*

*Initializes a new instance of the **System.Data.DBConcurrencyException** class.*

*You can create a new exception that catches an earlier exception. The code that handles the second exception can make use of the additional information from*

1 the earlier exception, also called an inner exception, to examine the cause of the  
2 initial error. The text string describing the details of the exception. A reference to  
3 an inner exception.

4 *HelpLink*

5 *HResult*

6 *InnerException*

7 *Message*

8 *Row*

9 *ToString*

12 *Description*

13 *Gets or sets the value of the System.Data.DataRow .*

14 *Use System.Data.DBConcurrencyException.Row to retrieve the value of*  
15 *the System.Data.DataRow row that generated the*

16 *System.Data.DBConcurrencyException . Setting the value of the*

17 *System.Data.DataRow has no effect.*

18 *Source*

19 *StackTrace*

20 *TargetSite*

21 *DbType enumeration (System.Data)*

22 *ToString*

25 *Description*

1 Gets the data type of a field, a property, or a **Parameter** object of a .NET  
2 data provider.

3 The type of a parameter is specific to the .NET data provider. Specifying  
4 the type converts the value of the **Parameter** to the .NET data provider Type  
5 before passing the value to the data source. If the type is not specified, ADO.NET  
6 infers the .NET data provider Type of the **Parameter** from the .NET Framework  
7 Type from the **Value** property of the **Parameter** object.

8 ToString

9  
10 [C#] public const DbType AnsiString;

11 [C++] public: const DbType AnsiString;

12 [VB] Public Const AnsiString As DbType

13 [JScript] public var AnsiString : DbType;

14  
15 Description

16 A variable-length stream of non-Unicode characters ranging between 1  
17 and 8,000 characters.

18 ToString

19  
20 [C#] public const DbType AnsiStringFixedLength;

21 [C++] public: const DbType AnsiStringFixedLength;

22 [VB] Public Const AnsiStringFixedLength As DbType

23 [JScript] public var AnsiStringFixedLength : DbType;

24 ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const DbType Binary;
[C++] public: const DbType Binary;
[VB] Public Const Binary As DbType
[JScript] public var Binary : DbType;
```

*Description*

*A variable-length stream of binary data ranging between 1 and 8,000 bytes.*

*ToString*

```
[C#] public const DbType Boolean;
[C++] public: const DbType Boolean;
[VB] Public Const Boolean As DbType
[JScript] public var Boolean : DbType;
```

*Description*

*A simple type representing Boolean values of **true** or **false** .*

*ToString*

```
[C#] public const DbType Byte;
[C++] public: const DbType Byte;
[VB] Public Const Byte As DbType
[JScript] public var Byte : DbType;
```

*Description*

1        *An 8-bit unsigned integer.*

2        *ToString*

3  
4        *[C#] public const DbType Currency;*

5        *[C++] public: const DbType Currency;*

6        *[VB] Public Const Currency As DbType*

7        *[JScript] public var Currency : DbType;*

8  
9        *Description*

10        *A currency value ranging from -2 (or -922,337,203,685,477.5808) to 2 -1*  
11        *(or +922,337,203,685,477.5807) with an accuracy to a ten-thousandth of a*  
12        *currency unit.*

13        *ToString*

14  
15        *[C#] public const DbType Date;*

16        *[C++] public: const DbType Date;*

17        *[VB] Public Const Date As DbType*

18        *[JScript] public var Date : DbType;*

19  
20        *Description*

21        *Date and time data ranging in value from January 1, 1753 to December 31,*  
22        *9999 to an accuracy of 3.33 milliseconds.*

23        *ToString*

24  
25        *[C#] public const DbType DateTime;*

1 *[C++] public: const DbType DateTime;*  
 2 *[VB] Public Const DateTime As DbType*  
 3 *[JScript] public var DateTime : DbType;*

4  
 5 *Description*

6 *A type representing a date and time value.*

7 *ToString*

8  
 9 *[C#] public const DbType Decimal;*  
 10 *[C++] public: const DbType Decimal;*  
 11 *[VB] Public Const Decimal As DbType*  
 12 *[JScript] public var Decimal : DbType;*

13  
 14 *Description*

15 *A simple type representing values ranging from 1.0 x 10 to approximately*  
 16 *7.9 x 10 with 28-29 significant digits.*

17 *ToString*

18  
 19 *[C#] public const DbType Double;*  
 20 *[C++] public: const DbType Double;*  
 21 *[VB] Public Const Double As DbType*  
 22 *[JScript] public var Double : DbType;*

23  
 24 *Description*

25

1        *A floating point type representing values ranging from approximately 5.0 x*  
2        *10 to 1.7 x 10 with a precision of 15-16 digits.*

3        *ToString*

4  
5        *[C#] public const DbType Guid;*

6        *[C++] public: const DbType Guid;*

7        *[VB] Public Const Guid As DbType*

8        *[JScript] public var Guid : DbType;*

9  
10       *Description*

11       *A globally unique identifier (or GUID).*

12       *ToString*

13  
14       *[C#] public const DbType Int16;*

15       *[C++] public: const DbType Int16;*

16       *[VB] Public Const Int16 As DbType*

17       *[JScript] public var Int16 : DbType;*

18  
19       *Description*

20       *An integral type representing signed 16-bit integers with values between -*  
21       *32768 and 32767.*

22       *ToString*

23  
24       *[C#] public const DbType Int32;*

25       *[C++] public: const DbType Int32;*

1 *[VB] Public Const Int32 As DbType*

2 *[JScript] public var Int32 : DbType;*

3

4 *Description*

5 *An integral type representing signed 32-bit integers with values between -*  
6 *2147483648 and 2147483647.*

7 *ToString*

8

9 *[C#] public const DbType Int64;*

10 *[C++] public: const DbType Int64;*

11 *[VB] Public Const Int64 As DbType*

12 *[JScript] public var Int64 : DbType;*

13

14 *Description*

15 *An integral type representing signed 64-bit integers with values between -*  
16 *9223372036854775808 and 9223372036854775807.*

17 *ToString*

18

19 *[C#] public const DbType Object;*

20 *[C++] public: const DbType Object;*

21 *[VB] Public Const Object As DbType*

22 *[JScript] public var Object : DbType;*

23

24 *Description*

25

1        *A general type representing any reference or value type not explicitly*  
2 *represented by another **TypeCode** .*

3        *ToString*

4  
5 *[C#] public const DbType SByte;*

6 *[C++] public: const DbType SByte;*

7 *[VB] Public Const SByte As DbType*

8 *[JScript] public var SByte : DbType;*

9  
10 *Description*

11        *An integral type representing signed 8-bit integers with values between -*  
12 *128 and 127.*

13        *ToString*

14  
15 *[C#] public const DbType Single;*

16 *[C++] public: const DbType Single;*

17 *[VB] Public Const Single As DbType*

18 *[JScript] public var Single : DbType;*

19  
20 *Description*

21        *A floating point type representing values ranging from approximately 1.5 x*  
22 *10 to 3.4 x 10 with a precision of 7 digits.*

23        *ToString*

24  
25 *[C#] public const DbType String;*

1 *[C++] public: const DbType String;*  
 2 *[VB] Public Const String As DbType*  
 3 *[JScript] public var String : DbType;*

4  
 5 *Description*

6 *A sealed class type representing Unicode character strings.*

7 *ToString*

8  
 9 *[C#] public const DbType StringFixedLength;*  
 10 *[C++] public: const DbType StringFixedLength;*  
 11 *[VB] Public Const StringFixedLength As DbType*  
 12 *[JScript] public var StringFixedLength : DbType;*

13 *ToString*

14  
 15 *[C#] public const DbType Time;*  
 16 *[C++] public: const DbType Time;*  
 17 *[VB] Public Const Time As DbType*  
 18 *[JScript] public var Time : DbType;*

19  
 20 *Description*

21 *Date and time data ranging in value from January 1, 1753 to December 31,*  
 22 *9999 to an accuracy of 3.33 milliseconds.*

23 *ToString*

24  
 25 *[C#] public const DbType UInt16;*

1 *[C++] public: const DbType UInt16;*  
 2 *[VB] Public Const UInt16 As DbType*  
 3 *[JScript] public var UInt16 : DbType;*

4  
 5 *Description*

6 *An integral type representing unsigned 16-bit integers with values between*  
 7 *0 and 65535.*

8 *ToString*

9  
 10 *[C#] public const DbType UInt32;*  
 11 *[C++] public: const DbType UInt32;*  
 12 *[VB] Public Const UInt32 As DbType*  
 13 *[JScript] public var UInt32 : DbType;*

14  
 15 *Description*

16 *An integral type representing unsigned 32-bit integers with values between*  
 17 *0 and 4294967295.*

18 *ToString*

19  
 20 *[C#] public const DbType UInt64;*  
 21 *[C++] public: const DbType UInt64;*  
 22 *[VB] Public Const UInt64 As DbType*  
 23 *[JScript] public var UInt64 : DbType;*

24  
 25 *Description*

*An integral type representing unsigned 64-bit integers with values between 0 and 18446744073709551615.*

*ToString*

*[C#] public const DbType VarNumeric;*

*[C++] public: const DbType VarNumeric;*

*[VB] Public Const VarNumeric As DbType*

*[JScript] public var VarNumeric : DbType;*

*DeletedRowInaccessibleException class (System.Data)*

*ToString*

### *Description*

*Represents the exception that is thrown when an action is attempted on a **System.Data.DataRow** that has been deleted.*

*To delete a **System.Data.DataRow** , use the **System.Data.DataRow** class's **System.Data.DataRow.Delete** method. Once you have deleted a row, any attempts to manipulate it will generate the **System.Data.DeletedRowInaccessibleException***

*DeletedRowInaccessibleException*

*Example Syntax:*

*ToString*

*[C#] public DeletedRowInaccessibleException();*

*[C++] public: DeletedRowInaccessibleException();*

1 *[VB] Public Sub New()*

2 *[JScript] public function DeletedRowInaccessibleException(); Initializes a new*  
 3 *instance of the **System.Data.DeletedRowInaccessibleException** class.*

4  
 5 *Description*

6 *Initializes a new instance of the*  
 7 ***System.Data.DeletedRowInaccessibleException** class.*

8 *Use the **System.Data.DataRow** class's **System.Data.DataRow.RowState** to*  
 9 *determine if a row has been deleted.*

10 *DeletedRowInaccessibleException*

11 *Example Syntax:*

12 *ToString*

13  
 14 *[C#] public DeletedRowInaccessibleException(string s);*

15 *[C++] public: DeletedRowInaccessibleException(String\* s);*

16 *[VB] Public Sub New(ByVal s As String)*

17 *[JScript] public function DeletedRowInaccessibleException(s : String);*

18  
 19 *Description*

20 *Initializes a new instance of the*  
 21 ***System.Data.DeletedRowInaccessibleException** class with the specified string.*

22 *Use the **System.Data.DataRow** class's **System.Data.DataRow.RowState** to*  
 23 *determine if a row has been deleted. The string to display when the exception is*  
 24 *thrown.*

25 *DeletedRowInaccessibleException*

*Example Syntax:*

*ToString*

*[C#] public DeletedRowInaccessibleException(SerializationInfo info,  
StreamingContext context);*  
*[C++] public: DeletedRowInaccessibleException(SerializationInfo\* info,  
StreamingContext context);*  
*[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As  
StreamingContext)*  
*[JScript] public function DeletedRowInaccessibleException(info :  
SerializationInfo, context : StreamingContext);* *Initializes a new instance of the  
System.Data.DeletedRowInaccessibleException class.*

*Description*

*Initializes a new instance of the  
System.Data.DeletedRowInaccessibleException class with serialization  
information. The data necessary to serialize or deserialize an object. Description  
of the source and destination of the specified serialized stream.*

*HelpLink*

*HResult*

*InnerException*

*Message*

*Source*

*StackTrace*

*TargetSite*

*DuplicateNameException class (System.Data)*

*ToString*

### *Description*

*Represents the exception that is thrown when a duplicate database object name is encountered during an add operation in a **System.Data.DataSet** -related object.*

*Examples of duplicate database object names that may be encountered are tables, columns, relations, or constraints.*

*DuplicateNameException*

*Example Syntax:*

*ToString*

*[C#] public DuplicateNameException();*

*[C++] public: DuplicateNameException();*

*[VB] Public Sub New()*

*[JScript] public function DuplicateNameException();*

### *Description*

*Initializes a new instance of the **System.Data.DuplicateNameException** class.*

*DuplicateNameException*

*Example Syntax:*

*ToString*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public DuplicateNameException(string s);
[C++] public: DuplicateNameException(String* s);
[VB] Public Sub New(ByVal s As String)
[JScript] public function DuplicateNameException(s : String);
```

*Description*

*Initializes a new instance of the **System.Data.DuplicateNameException** class with the specified string. The string to display when the exception is thrown.*

*DuplicateNameException*

*Example Syntax:*

*ToString*

```
[C#] public DuplicateNameException(SerializationInfo info, StreamingContext
context);
[C++] public: DuplicateNameException(SerializationInfo* info,
StreamingContext context);
[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As
StreamingContext)
[JScript] public function DuplicateNameException(info : SerializationInfo,
context : StreamingContext); Initializes a new instance of the
System.Data.DuplicateNameException class.
```

*Description*

1        *Initializes a new instance of the **System.Data.DuplicateNameException***  
2 *class with serialization information. The data necessary to serialize or deserialize*  
3 *an object. Description of the source and destination of the specified serialized*  
4 *stream.*

5        *HelpLink*

6        *HResult*

7        *InnerException*

8        *Message*

9        *Source*

10       *StackTrace*

11       *TargetSite*

12       *EvaluateException class (System.Data)*

13       *ToString*

14  
15  
16       *Description*

17       *Represents the exception that is thrown when the*  
18 ***System.Data.DataColumn.Expression** property of a **System.Data.DataColumn***  
19 *cannot be evaluated.*

20       *EvaluateException*

21       *Example Syntax:*

22       *ToString*

23  
24       *[C#] public EvaluateException();*

25       *[C++] public: EvaluateException();*

1 *[VB] Public Sub New()*

2 *[JScript] public function EvaluateException(); Initializes a new instance of the*  
 3 *System.Data.EvaluateException class.*

4  
 5 *Description*

6 *Initializes a new instance of the System.Data.EvaluateException class.*

7 *EvaluateException*

8 *Example Syntax:*

9 *ToString*

10  
 11 *[C#] public EvaluateException(string s);*

12 *[C++] public: EvaluateException(String\* s);*

13 *[VB] Public Sub New(ByVal s As String)*

14 *[JScript] public function EvaluateException(s : String);*

15  
 16 *Description*

17 *Initializes a new instance of the System.Data.EvaluateException class with*  
 18 *the specified string. The string to display when the exception is thrown.*

19 *EvaluateException*

20 *Example Syntax:*

21 *ToString*

22  
 23 *[C#] public EvaluateException(SerializationInfo info, StreamingContext context);*

24 *[C++] public: EvaluateException(SerializationInfo\* info, StreamingContext*  
 25 *context);*

1 *[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As*  
2 *StreamingContext)*

3 *[JScript] public function EvaluateException(info : SerializationInfo, context :*  
4 *StreamingContext);*

5  
6 *Description*

7 *Initializes a new instance of the **System.Data.EvaluateException** class with*  
8 *the **System.Runtime.Serialization.SerializationInfo** and the*  
9 ***System.Runtime.Serialization.StreamingContext** . The data needed to serialize or*  
10 *deserialize an object. The source and destination of a given serialized stream.*

11 *HelpLink*

12 *HResult*

13 *InnerException*

14 *Message*

15 *Source*

16 *StackTrace*

17 *TargetSite*

18 *FillErrorEventArgs class (System.Data)*

19 *ToString*

20  
21  
22 *Description*

23 *Provides data for the **System.Data.Common.DbDataAdapter.FillError***  
24 *event of a **System.Data.Common.DbDataAdapter** .*

The data is used by the **System.Data.Common.DbDataAdapter.OnFillError(System.Data.FillErrorEventArgs)** method of the **System.Data.Common.DbDataAdapter** .

**FillErrorEventArgs**

*Example Syntax:*

*ToString*

[C#] public FillErrorEventArgs(DataTable dataTable, object[] values);

[C++] public: FillErrorEventArgs(DataTable\* dataTable, Object\* values \_\_gc[]);

[VB] Public Sub New(ByVal dataTable As DataTable, ByVal values() As Object)

[JScript] public function FillErrorEventArgs(dataTable : DataTable, values : Object[]);

## Description

Initializes a new instance of the **System.Data.FillErrorEventArgs** class.

The **System.Data.DataTable** being updated. The values for the row being updated.

*Continue*

*ToString*

[C#] public bool Continue {get; set;}

[C++] public: \_\_property bool get\_Continue();public: \_\_property void set\_Continue(bool);

[VB] Public Property Continue As Boolean

[JScript] public function get Continue() : Boolean;public function set

1 *Continue(Boolean);*

2  
3 *Description*

4 *Gets or sets a value indicating whether to continue the fill operation despite*  
5 *the error.*

6 *DataTable*

7 *ToString*

8  
9 *[C#] public DataTable DataTable {get;}*

10 *[C++] public: \_\_property DataTable\* get\_DataTable();*

11 *[VB] Public ReadOnly Property DataTable As DataTable*

12 *[JScript] public function get DataTable() : DataTable;*

13  
14 *Description*

15 *Gets the **System.Data.DataTable** being updated when the error occurred.*

16 *Errors*

17 *ToString*

18  
19 *[C#] public Exception Errors {get; set;}*

20 *[C++] public: \_\_property Exception\* get\_Errors();public: \_\_property void*  
21 *set\_Errors(Exception\*);*

22 *[VB] Public Property Errors As Exception*

23 *[JScript] public function get Errors() : Exception;public function set*  
24 *Errors(Exception);*

## Description

*Gets the errors being handled.*

*Values*

*ToString*

*[C#] public object[] Values {get;}*

*[C++] public: \_\_property Object\* get\_Values();*

*[VB] Public ReadOnly Property Values As Object ()*

*[JScript] public function get Values() : Object[];*

## Description

*Gets the values for the row being updated when the error occurred.*

*FillErrorHandler delegate (System.Data)*

*ToString*

## Description

*Represents the method that will handle the*

***System.Data.Common.DbDataAdapter.FillError** event. The source of the event.*

*The **System.Data.FillEventArgs** that contains the event data.*

*When you create a **System.Data.FillErrorHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is*

called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

*ForeignKeyConstraint class (System.Data)*

*ToString*

### *Description*

*Represents an action restriction enforced on a set of columns in a primary key/foreign key relationship when a value or row is either deleted or updated.*

*A System.Data.ForeignKeyConstraint restricts the action performed when a value in a column (or columns) is either deleted or updated. Such a constraint is intended to be used with primary key columns. In a parent/child relationship between two tables, deleting a value from the parent table can affect the child rows in one of the following ways.*

*ForeignKeyConstraint*

*Example Syntax:*

*ToString*

*[C#] public ForeignKeyConstraint(DataColumn parentColumn, DataColumn childColumn);*

*[C++] public: ForeignKeyConstraint(DataColumn\* parentColumn, DataColumn\* childColumn);*

*[VB] Public Sub New(ByVal parentColumn As DataColumn, ByVal childColumn As DataColumn)*

*[JScript] public function ForeignKeyConstraint(parentColumn : DataColumn,*

1 *childColumn : DataColumn); Initializes a new instance of the*  
2 *System.Data.ForeignKeyConstraint class.*

3  
4 *Description*

5 *Initializes a new instance of the System.Data.ForeignKeyConstraint class*  
6 *with the specified parent and child System.Data.DataColumn objects. The parent*  
7 *System.Data.DataColumn in the constraint. The child System.Data.DataColumn*  
8 *in the constraint.*

9 *ForeignKeyConstraint*

10 *Example Syntax:*

11 *ToString*

12  
13 *[C#] public ForeignKeyConstraint(DataColumn[] parentColumns, DataColumn[]*  
14 *childColumns);*

15 *[C++] public: ForeignKeyConstraint(DataColumn\* parentColumns[],*  
16 *DataColumn\* childColumns[]);*

17 *[VB] Public Sub New(ByVal parentColumns() As DataColumn, ByVal*  
18 *childColumns() As DataColumn)*

19 *[JScript] public function ForeignKeyConstraint(parentColumns : DataColumn[],*  
20 *childColumns : DataColumn[]);*

21  
22 *Description*

23 *Initializes a new instance of the System.Data.ForeignKeyConstraint class*  
24 *with the specified arrays of parent and child System.Data.DataColumn objects.*

1 *An array of parent **System.Data.DataColumn** in the constraint. An array of child*  
2 ***System.Data.DataColumn** in the constraint.*

3 *ForeignKeyConstraint*

4 *Example Syntax:*

5 *ToString*

6  
7 *[C#] public ForeignKeyConstraint(string constraintName, DataColumn*  
8 *parentColumn, DataColumn childColumn);*

9 *[C++] public: ForeignKeyConstraint(String\* constraintName, DataColumn\**  
10 *parentColumn, DataColumn\* childColumn);*

11 *[VB] Public Sub New(ByVal constraintName As String, ByVal parentColumn As*  
12 *DataColumn, ByVal childColumn As DataColumn)*

13 *[JScript] public function ForeignKeyConstraint(constraintName : String,*  
14 *parentColumn : DataColumn, childColumn : DataColumn);*

15  
16 *Description*

17 *Initializes a new instance of the **System.Data.ForeignKeyConstraint** class*  
18 *with the specified name, parent and child **System.Data.DataColumn** objects. The*  
19 *name of the constraint. The parent **System.Data.DataColumn** in the constraint.*  
20 *The child **System.Data.DataColumn** in the constraint.*

21 *ForeignKeyConstraint*

22 *Example Syntax:*

23 *ToString*

24  
25 *[C#] public ForeignKeyConstraint(string constraintName, DataColumn[]*

```

1  parentColumns, DataColumn[] childColumns);
2  [C++] public: ForeignKeyConstraint(String* constraintName, DataColumn*
3  parentColumns[], DataColumn* childColumns[]);
4  [VB] Public Sub New(ByVal constraintName As String, ByVal parentColumns()
5  As DataColumn, ByVal childColumns() As DataColumn)
6  [JScript] public function ForeignKeyConstraint(constraintName : String,
7  parentColumns : DataColumn[], childColumns : DataColumn[]);
8

```

### Description

Initializes a new instance of the **System.Data.ForeignKeyConstraint** class with the specified name, and arrays of parent and child **System.Data.DataColumn** objects. The name of the **System.Data.ForeignKeyConstraint**. If null or empty string, a default name will be given when added to the constraints collection. An array of parent **System.Data.DataColumn** in the constraint. An array of child **System.Data.DataColumn** in the constraint.

### ForeignKeyConstraint

#### Example Syntax:

#### ToString

```

20 [C#] public ForeignKeyConstraint(string constraintName, string
21 parentTableName, string[] parentColumnNames, string[] childColumnNames,
22 AcceptRejectRule acceptRejectRule, Rule deleteRule, Rule updateRule);
23 [C++] public: ForeignKeyConstraint(String* constraintName, String*
24 parentTableName, String* parentColumnNames __gc[], String*
25 childColumnNames __gc[], AcceptRejectRule acceptRejectRule, Rule deleteRule,

```

```

1 Rule updateRule);
2 [VB] Public Sub New(ByVal constraintName As String, ByVal parentTableName
3 As String, ByVal parentColumnNames() As String, ByVal childColumnNames() As
4 String, ByVal acceptRejectRule As AcceptRejectRule, ByVal deleteRule As Rule,
5 ByVal updateRule As Rule)
6 [JScript] public function ForeignKeyConstraint(constraintName : String,
7 parentTableName : String, parentColumnNames : String[], childColumnNames :
8 String[], acceptRejectRule : AcceptRejectRule, deleteRule : Rule, updateRule :
9 Rule);

```

#### Description

Initializes a new instance of the **System.Data.ForeignKeyConstraint** class with the specified name, and arrays of parent and child **System.Data.DataColumn** objects, the parent **System.Data.DataTable** name, and various rule settings. The name of the constraint. The names of the parent **System.Data.DataTable** that contains parent **System.Data.DataColumn** objects in the constraint. An array of the names of parent **System.Data.DataColumn** objects in the constraint. An array of the names of child **System.Data.DataColumn** objects in the constraint. One of the **System.Data.AcceptRejectRule** values. Possible values include **None**, **Cascade**, and **Default**. One of the **System.Data.Rule** values to use when a row is deleted. The default is **Cascade**. Possible values include: **None**, **Cascade**, **SetNull**, **SetDefault**, and **Default**. One of the **System.Data.Rule** values to use when a row is updated. The default is **Cascade**. Possible values include: **None**, **Cascade**, **SetNull**, **SetDefault**, and **Default**.

DataSet

*AcceptRejectRule*

*ToString*

*Description*

*Indicates the action that should take place across this constraint when **System.Data.DataTable.AcceptChanges** is invoked.*

*Changes to a **System.Data.DataRow** or **System.Data.DataTable** are not final until the **AcceptChanges** method is invoked. At that point, the **System.Data.ForeignKeyConstraint.AcceptRejectRule** determines the course of action on any values that have been changed or deleted.*

*Columns*

*ToString*

*[C#] public virtual DataColumn[] Columns {get;}*

*[C++] public: \_\_property virtual DataColumn\* get\_Columns();*

*[VB] Overridable Public ReadOnly Property Columns As DataColumn ()*

*[JScript] public function get Columns() : DataColumn[];*

*Description*

*Gets the child columns of this constraint.*

*ConstraintName*

*DeleteRule*

*ToString*

### *Description*

*Gets or sets the action that occurs across this constraint when a row is deleted.*

*When a row is deleted from a parent table, the **System.Data.ForeignKeyConstraint.DeleteRule** determines what will happen in the columns of the child table (or tables). If the rule is set to **Cascade**, child rows will be deleted.*

### *ExtendedProperties*

### *RelatedColumns*

### *ToString*

### *Description*

*The parent columns of this constraint.*

### *RelatedTable*

### *ToString*

*[C#] public virtual DataTable RelatedTable {get;}*

*[C++] public: \_\_property virtual DataTable\* get\_RelatedTable();*

*[VB] Overridable Public ReadOnly Property RelatedTable As DataTable*

*[JScript] public function get RelatedTable() : DataTable;*

### *Description*

*Gets the parent table of this constraint.*

*Table*

*ToString*

*[C#] public override DataTable Table {get;}*

*[C++] public: \_\_property virtual DataTable\* get\_Table();*

*[VB] Overrides Public ReadOnly Property Table As DataTable*

*[JScript] public function get Table() : DataTable;*

#### *Description*

*Gets the child table of this constraint.*

*UpdateRule*

*ToString*

*[C#] public virtual Rule UpdateRule {get; set;}*

*[C++] public: \_\_property virtual Rule get\_UpdateRule();public: \_\_property*

*virtual void set\_UpdateRule(Rule);*

*[VB] Overridable Public Property UpdateRule As Rule*

*[JScript] public function get UpdateRule() : Rule;public function set*

*UpdateRule(Rule);*

#### *Description*

*Gets or sets the action that occurs across this constraint on when a row is updated.*

*Equals*

1 .  
2 *[C#] public override bool Equals(object key);*  
3 *[C++] public: bool Equals(Object\* key);*  
4 *[VB] Overrides Public Function Equals(ByVal key As Object) As Boolean*  
5 *[JScript] public override function Equals(key : Object) : Boolean;*

#### 6 7 *Description*

8 *Gets a value indicating whether the current*  
9 ***System.Data.ForeignKeyConstraint** is identical to the specified object.*  
10 *Return Value: **true** , if the objects are identical; otherwise, **false** . The object to*  
11 *which this **System.Data.ForeignKeyConstraint** is compared. Two*  
12 ***System.Data.ForeignKeyConstraint** are equal if they constrain the same columns.*

#### 13 *GetHashCode*

14  
15 *[C#] public override int GetHashCode();*  
16 *[C++] public: int GetHashCode();*  
17 *[VB] Overrides Public Function GetHashCode() As Integer*  
18 *[JScript] public override function GetHashCode() : int;*

#### 19 20 *Description*

21 *Gets the hash code of this instance of the*  
22 ***System.Data.ForeignKeyConstraint** object.*  
23 *Return Value: A 32-bit signed integer hash code.*

#### 24 *IColumnMapping interface (System.Data)*

#### 25 *ToString*

1  
2  
3 *Description*

4 *Associates a data source column with a **System.Data.DataSet** column, and*  
5 *is implemented by the **System.Data.Common.DataColumnMapping** class, which*  
6 *is used in common by .NET data providers.*

7 *The **System.Data.IColumnMapping** interface allows an inheriting class to*  
8 *implement a **ColumnMapping** class, which associates a data source column with a*  
9 ***System.Data.DataSet** column. For more information, see .*

10 *DataSetColumn*

11 *ToString*

12  
13 *[C#] string DataSetColumn {get; set;}*

14 *[C++] String\* get\_DataSetColumn();void set\_DataSetColumn(String\*);*

15 *[VB] Property DataSetColumn As String*

16 *[JScript] abstract function get DataSetColumn() : String;public abstract function*  
17 *set DataSetColumn(String);*

18  
19 *Description*

20 *Gets or sets the name of the column within the **System.Data.DataSet** to map*  
21 *to.*

22 *SourceColumn*

23 *ToString*

24  
25 *[C#] string SourceColumn {get; set;}*

1 *[C++] String\* get\_SourceColumn();void set\_SourceColumn(String\*);*  
 2 *[VB] Property SourceColumn As String*  
 3 *[JScript] abstract function get SourceColumn() : String;public abstract function*  
 4 *set SourceColumn(String);*

5  
 6 *Description*

7 *Gets or sets the case-sensitive column name from a data source to map*  
 8 *from.*

9 *ICollectionMappingCollection interface (System.Data)*

10 *ToString*

11  
 12  
 13 *Description*

14 *Contains a collection of ColumnMapping objects, and is implemented by*  
 15 *the **System.Data.Common.DataColumnMappingCollection** , which is used in*  
 16 *common by .NET data providers.*

17 *The **System.Data.ICollectionMappingCollection** interface allows an*  
 18 *inheriting class to implement a ColumnMapping collection. For more information,*  
 19 *see .*

20 *Item*

21 *ToString*

22  
 23 *[C#] object this[string index] {get; set;}*

24 *[C++] Object\* get\_Item(String\* index);void set\_Item(String\* index, Object\*);*

25 *[VB] Default Property Item(ByVal index As String) As Object*

1 *[JScript] abstract returnValue =*  
 2 *IColumnMappingCollectionObject.Item(index);IColumnMappingCollectionObject*  
 3 *.Item(index) = returnValue;*

4  
 5 *Description*

6 *Gets or sets the System.Data.Common.DataColumnMapping object with*  
 7 *the specified name. The name of the System.Data.Common.DataColumnMapping*  
 8 *object to find.*

9 *Add*

10  
 11 *[C#] IColumnMapping Add(string sourceColumnName, string*  
 12 *dataSetColumnName);*

13 *[C++] IColumnMapping\* Add(String\* sourceColumnName, String\**  
 14 *dataSetColumnName);*

15 *[VB] Function Add(ByVal sourceColumnName As String, ByVal*  
 16 *dataSetColumnName As String) As IColumnMapping*

17 *[JScript] function Add(sourceColumnName : String, dataSetColumnName :*  
 18 *String) : IColumnMapping;*

19  
 20 *Description*

21 *Adds a System.Data.Common.DataColumnMapping to the*  
 22 *System.Data.Common.DataColumnMappingCollection using the source column*  
 23 *and System.Data.DataSet column names.*

24 *Return Value: A reference to the newly-mapped*

25

***System.Data.Common.DataColumnMapping*** object. The case-sensitive name of the source column. The name of the ***System.Data.DataSet*** column.

*Contains*

***[C#] bool Contains(string sourceColumnName);***

***[C++] bool Contains(String\* sourceColumnName);***

***[VB] Function Contains(ByVal sourceColumnName As String) As Boolean***

***[JScript] function Contains(sourceColumnName : String) : Boolean;***

*Description*

*Gets a value indicating whether the ***System.Data.Common.DataColumnMappingCollection*** contains a ***System.Data.Common.DataColumnMapping*** with the specified source column name.*

*Return Value: true if a ***System.Data.Common.DataColumnMapping*** with the specified source column name exists, otherwise ***false*** . The case-sensitive name of the source column.*

*GetByDataSetColumn*

***[C#] IColumnMapping GetByDataSetColumn(string dataSetColumnName);***

***[C++] IColumnMapping\* GetByDataSetColumn(String\* dataSetColumnName);***

***[VB] Function GetByDataSetColumn(ByVal dataSetColumnName As String) As IColumnMapping***

***[JScript] function GetByDataSetColumn(dataSetColumnName : String) :***

***IColumnMapping;***

## *Description*

*Gets a reference to a **System.Data.Common.DataColumnMapping** using the name of the **System.Data.DataSet** column.*

*Return Value: A reference to a **System.Data.Common.DataColumnMapping**.*

*The name of the **System.Data.DataSet** column within the collection.*

### *IndexOf*

*[C#] int IndexOf(string sourceColumnName);*

*[C++] int IndexOf(String\* sourceColumnName);*

*[VB] Function IndexOf(ByVal sourceColumnName As String) As Integer*

*[JScript] function IndexOf(sourceColumnName : String) : int;*

## *Description*

*Gets the location of the **System.Data.Common.DataColumnMapping** with the specified source column name.*

*Return Value: The location of the **System.Data.Common.DataColumnMapping** with the specified case-sensitive source column name. The case-sensitive name of the source column.*

### *RemoveAt*

*[C#] void RemoveAt(string sourceColumnName);*

*[C++] void RemoveAt(String\* sourceColumnName);*

*[VB] Sub RemoveAt(ByVal sourceColumnName As String)*

*[JScript] function RemoveAt(sourceColumnName : String);*

## *Description*

*Removes the **System.Data.Common.DataColumnMapping** object with the specified source column name from the collection. The case-sensitive source column name.*

*IDataAdapter interface (System.Data)*

*RemoveAt*

## *Description*

*Allows an object to implement a **DataAdapter**, and represents a set of methods and mapping action-related properties used to fill and refresh a **System.Data.DataSet** and update a data source.*

*The **System.Data.IDataAdapter** interface allows an inheriting class to implement a **DataAdapter** class, which represents the bridge between a data source and a **System.Data.DataSet** . For more information about **DataAdapter** classes, see . For more information about implementing .NET data providers, see .*

*MissingMappingAction*

*RemoveAt*

*[C#] MissingMappingAction MissingMappingAction {get; set;}*

*[C++] MissingMappingAction get\_MissingMappingAction();void  
set\_MissingMappingAction(MissingMappingAction);*

*[VB] Property MissingMappingAction As MissingMappingAction*

*[JScript] abstract function get MissingMappingAction() :*

1 *MissingMappingAction*;public abstract function set  
2 *MissingMappingAction*(*MissingMappingAction*);

3  
4 *Description*

5 *Indicates or specifies whether unmapped source tables or columns are*  
6 *passed with their source names in order to be filtered or to raise an error.*

7 *The **System.Data.IDataAdapter.TableMappings** property provides the*  
8 *master mapping between the returned records and the **System.Data.DataSet** .*

9 *MissingSchemaAction*

10 *RemoveAt*

11  
12 [C#] *MissingSchemaAction MissingSchemaAction {get; set;}*

13 [C++] *MissingSchemaAction get\_MissingSchemaAction();void*  
14 *set\_MissingSchemaAction(MissingSchemaAction);*

15 [VB] *Property MissingSchemaAction As MissingSchemaAction*

16 [JScript] *abstract function get MissingSchemaAction() :*

17 *MissingSchemaAction*;public abstract function set  
18 *MissingSchemaAction*(*MissingSchemaAction*);

19  
20 *Description*

21 *Indicates or specifies whether missing source tables, columns, and their*  
22 *relationships are added to the data set schema, ignored, or cause an error to be*  
23 *raised.*

24 *TableMappings*

25 *RemoveAt*

```

1
2 [C#] ITableMappingCollection TableMappings {get;}
3 [C++] ITableMappingCollection* get_TableMappings();
4 [VB] ReadOnly Property TableMappings As ITableMappingCollection
5 [JScript] abstract function get TableMappings() : ITableMappingCollection;
6

```

### *Description*

*Indicates how a source table is mapped to a data set table.*

*The **System.Data.IDataAdapter** uses only the mappings for the source table named "Table". All SELECT, INSERT, DELETE, and UPDATE statements returning data must do so using consistent column naming. The column names returned in the records must be unique, otherwise columns with duplicate names overwrite previous data. On **System.Data.IDataAdapter.Update(System.Data.DataSet)** , only the table mapped to the source table named "Table" will have its changes reconciled.*

### *Fill*

```

17
18 [C#] int Fill(DataSet dataSet);
19 [C++] int Fill(DataSet* dataSet);
20 [VB] Function Fill(ByVal dataSet As DataSet) As Integer
21 [JScript] function Fill(dataSet : DataSet) : int;
22

```

### *Description*

*Adds or refreshes rows in the **System.Data.DataSet** to match those in the data source using the **System.Data.DataSet** name, and creates a*

1 *System.Data.DataTable* named "Table".

2 *Return Value:* The number of rows successfully added to or refreshed in the  
3 *System.Data.DataSet*. This does not include rows affected by statements that do  
4 not return rows.

5 *System.Data.IDataAdapter.Fill(System.Data.DataSet)* retrieves rows from  
6 the data source using the *SELECT* statement specified by an associated  
7 *System.Data.IDbDataAdapter.SelectCommand* property. The connection object  
8 associated with the *SELECT* statement must be valid, but it does not need to be  
9 open. If the connection is closed before  
10 *System.Data.IDataAdapter.Fill(System.Data.DataSet)* is called, it is opened to  
11 retrieve data, then closed. If the connection is open before  
12 *System.Data.IDataAdapter.Fill(System.Data.DataSet)* is called, it remains open.  
13 A *System.Data.DataSet* to fill with records and, if necessary, schema.

#### 14 *FillSchema*

15  
16 [C#] *DataTable[] FillSchema(DataSet dataSet, SchemaType schemaType);*  
17 [C++] *DataTable\* FillSchema(DataSet\* dataSet, SchemaType schemaType) [];*  
18 [VB] *Function FillSchema(ByVal dataSet As DataSet, ByVal schemaType As*  
19 *SchemaType) As DataTable()*  
20 [JScript] *function FillSchema(dataSet : DataSet, schemaType : SchemaType) :*  
21 *DataTable[];*

#### 22 *Description*

23  
24 *Adds a System.Data.DataTable named "Table" to the specified*  
25 *System.Data.DataSet and configures the schema to match that in the data source*

1 *based on the specified **System.Data.SchemaType** .*

2 *Return Value: An array of **System.Data.DataTable** objects that contain schema*  
3 *information returned from the data source.*

4 *The*

5 ***System.Data.IDataAdapter.FillSchema(System.Data.DataSet, System.Data.Sche***  
6 ***maType)** method retrieves the schema from the data source using the*

7 ***System.Data.IDbDataAdapter.SelectCommand** . The connection object*  
8 *associated with the **System.Data.IDbDataAdapter.SelectCommand** must be valid,*  
9 *but it does not need to be open. If the connection is closed before*

10 ***System.Data.IDataAdapter.FillSchema(System.Data.DataSet, System.Data.Sche***  
11 ***maType)** is called, it is opened to retrieve data, then closed. If the connection is*  
12 *open before*

13 ***System.Data.IDataAdapter.FillSchema(System.Data.DataSet, System.Data.Sche***  
14 ***maType)** is called, it remains open. The **System.Data.DataSet** to be filled with the*  
15 *schema from the data source. One of the **System.Data.SchemaType** values.*

16 *GetFillParameters*

17  
18 *[C#] **IDataParameter[] GetFillParameters();***

19 *[C++] **IDataParameter\* GetFillParameters() [];***

20 *[VB] **Function GetFillParameters() As IDataParameter()***

21 *[JScript] **function GetFillParameters() : IDataParameter[];***

22  
23 *Description*

24 *Gets the parameters set by the user when executing an SQL SELECT*  
25 *statement.*

*Return Value:* An array of **System.Data.IDataParameter** objects that contains the parameters set by the user.

### *Update*

*[C#] int Update(DataSet dataSet);*

*[C++] int Update(DataSet\* dataSet);*

*[VB] Function Update(ByVal dataSet As DataSet) As Integer*

*[JScript] function Update(dataSet : DataSet) : int;*

### *Description*

*Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the specified **System.Data.DataSet** from a **System.Data.DataTable** named "Table".*

*Return Value:* The number of rows successfully updated from the **System.Data.DataSet**.

*When an application calls the **System.Data.IDataAdapter.Update(System.Data.DataSet)** method, the **System.Data.IDataAdapter** examines the **System.Data.DataRow.RowState** property, and executes the required INSERT, UPDATE, or DELETE statements based on the order of the indexes configured in the **System.Data.DataSet**. For example, **System.Data.IDataAdapter.Update(System.Data.DataSet)** might execute a DELETE statement, followed by an INSERT statement, and then another DELETE statement, due to the ordering of the rows in the **System.Data.DataTable**. An application can call the **System.Data.DataSet.GetChanges** method in situations where you must control the sequence of statement types (for example,*

*INSERTs before UPDATES). For more information, see . The **System.Data.DataSet** used to update the data source.*

***IDataParameter** interface (**System.Data**)*

***Update***

#### *Description*

*Represents a parameter to a **Command** object, and optionally, its mapping to **System.Data.DataSet** columns; and is implemented by .NET data providers that access data sources.*

*The **System.Data.IDataParameter** interface allows an inheriting class to implement a **Parameter** class, which represents a parameter to a **Command** object. For more information about **Parameter** classes, see . For more information about implementing .NET data providers, see .*

***DbType***

***Update***

*[C#] **DbType DbType** {get; set;}*

*[C++] **DbType** get\_DbType();void set\_DbType(DbType);*

*[VB] **Property DbType As DbType***

*[JScript] **abstract function** get DbType() : DbType;public **abstract function** set DbType(DbType);*

#### *Description*

***Gets or sets the System.Data.DbType of the parameter.***

1       The *PrvDbType* (where *Prv* represents the provider-specific prefix) and  
2       ***System.Data.SqlClient.SqlParameter.DbType*** are linked. Therefore, setting the  
3       ***System.Data.SqlClient.SqlParameter.DbType*** changes the *PrvDbType* to a  
4       supporting *PrvDbType*.

5       *Direction*

6       *Update*

7  
8       [C#] *ParameterDirection Direction {get; set;}*

9       [C++] *ParameterDirection get\_Direction();void*

10       *set\_Direction(ParameterDirection);*

11       [VB] *Property Direction As ParameterDirection*

12       [JScript] *abstract function get Direction() : ParameterDirection;public abstract*

13       *function set Direction(ParameterDirection);*

14  
15       *Description*

16       Gets or sets a value indicating whether the parameter is input-only, output-  
17       only, bidirectional, or a stored procedure return value parameter.

18       If the ***System.Data.ParameterDirection*** is output, and execution of the  
19       associated ***System.Data.SqlClient.SqlCommand*** does not return a value, the  
20       ***System.Data.IDataParameter*** contains a null value.

21       *IsNullable*

22       *Update*

23  
24       [C#] *bool IsNullable {get;}*

25       [C++] *bool get\_IsNullable();*

1 *[VB] ReadOnly Property IsNullable As Boolean*

2 *[JScript] abstract function get IsNullable() : Boolean;*

4 *Description*

5 *Gets or sets a value indicating whether the parameter accepts null values.*

6 *Null values are handled using the **System.DBNull** class.*

7 *ParameterName*

8 *Update*

10 *[C#] string ParameterName {get; set;}*

11 *[C++] String\* get\_ParameterName();void set\_ParameterName(String\*);*

12 *[VB] Property ParameterName As String*

13 *[JScript] abstract function get ParameterName() : String;public abstract function*  
14 *set ParameterName(String);*

16 *Description*

17 *Gets or sets the name of the **System.Data.IDataParameter** .*

18 *The **System.Data.IDataParameter.ParameterName** is specified in the form*

19 *@paramname. You must set **System.Data.IDataParameter.ParameterName***  
20 *before executing a command that relies on parameters.*

21 *SourceColumn*

22 *Update*

24 *[C#] string SourceColumn {get; set;}*

25 *[C++] String\* get\_SourceColumn();void set\_SourceColumn(String\*);*

1 *[VB] Property SourceColumn As String*

2 *[JScript] abstract function get SourceColumn() : String;public abstract function*  
3 *set SourceColumn(String);*

4  
5 *Description*

6 *Gets or sets the name of the source column that is mapped to the*  
7 *System.Data.DataSet and used for loading or returning the*  
8 *System.Data.IDataParameter.Value .*

9 *The link between the value of the System.Data.IDataParameter and the*  
10 *System.Data.DataTable may be bidirectional depending on the value of the*  
11 *System.Data.IDataParameter.Direction property.*

12 *SourceVersion*

13 *Update*

14  
15 *[C#] DataRowVersion SourceVersion {get; set;}*

16 *[C++] DataRowVersion get\_SourceVersion();void*  
17 *set\_SourceVersion(DataRowVersion);*

18 *[VB] Property SourceVersion As DataRowVersion*

19 *[JScript] abstract function get SourceVersion() : DataRowVersion;public abstract*  
20 *function set SourceVersion(DataRowVersion);*

21  
22 *Description*

23 *Gets or sets the System.Data.DataRowVersion to use when loading*  
24 *System.Data.IDataParameter.Value .*

This property is used by the **System.Data.IDbDataAdapter.UpdateCommand** during the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to determine whether the original or current value is used for a parameter value. This allows primary keys to be updated. This property is ignored by the **System.Data.IDbDataAdapter.InsertCommand** and **System.Data.IDbDataAdapter.DeleteCommand**. This property is set to the version of the **System.Data.DataRow** used by the **System.Data.DataRow.Item(System.Int32)** property, or the **System.Data.DataRow.GetChildRows(System.String)** method of the **System.Data.DataRow** object.

*Value*

*Update*

[C#] object Value {get; set;}

[C++] Object\* get\_Value();void set\_Value(Object\*);

[VB] Property Value As Object

[JScript] abstract function get Value() : Object;public abstract function set Value(Object);

#### *Description*

*Gets or sets the value of the parameter.*

*For input parameters, the value is bound to the **System.Data.IDbCommand** that is sent to the server. For output and return value parameters, the value is set*

on completion of the **System.Data.IDbCommand** and after the **System.Data.IDataReader** is closed.

*IDataParameterCollection* interface (System.Data)

*Update*

### *Description*

Collects all parameters relevant to a *Command* object and their mappings to **System.Data.DataSet** columns, and is implemented by .NET data providers that access data sources.

The **System.Data.IDataParameterCollection** interface allows an inheriting class to implement a *Parameter* collection. For more information about *Parameter* classes, see . For more information about implementing .NET data providers, see .

*Item*

*Update*

[C#] object this[string parameterName] {get; set;}

[C++] Object\* get\_Item(String\* parameterName); void set\_Item(String\* parameterName, Object\*);

[VB] Default Property Item(ByVal parameterName As String) As Object

[JScript] abstract returnValue =

*IDataParameterCollectionObject*.Item(parameterName); *IDataParameterCollectionObject*.Item(parameterName) = returnValue;

## Description

Gets the parameter at the specified index. The name of the parameter to retrieve.

## Contains

[C#] *bool Contains(string parameterName);*

[C++] *bool Contains(String\* parameterName);*

[VB] *Function Contains(ByVal parameterName As String) As Boolean*

[JScript] *function Contains(parameterName : String) : Boolean;*

## Description

Gets a value indicating whether a parameter in the collection has the specified source table name.

**Return Value:** *true* if the collection contains the parameter; otherwise, *false*. The name of the parameter.

## IndexOf

[C#] *int IndexOf(string parameterName);*

[C++] *int IndexOf(String\* parameterName);*

[VB] *Function IndexOf(ByVal parameterName As String) As Integer*

[JScript] *function IndexOf(parameterName : String) : int;*

## Description

Gets the location of the **System.Data.IDataParameter** within the collection.

**Return Value:** The location of the **System.Data.IDataParameter** within the collection. The name of the parameter.

**RemoveAt**

[C#] void RemoveAt(string parameterName);

[C++] void RemoveAt(String\* parameterName);

[VB] Sub RemoveAt(ByVal parameterName As String)

[JScript] function RemoveAt(parameterName : String);

**Description**

Removes the **System.Data.IDataParameter** from the collection. The name of the parameter.

**IDataReader** interface (System.Data)

**RemoveAt**

**Description**

Provides a means of reading one or more forward-only streams of result sets obtained by executing a command at a data source, and is implemented by .NET data providers that access relational databases.

The **System.Data.IDataReader** and **System.Data.IDataRecord** interfaces allow an inheriting class to implement a **DataReader** class, which provides a means of reading one or more forward-only streams of result sets. For more

information about *DataReader* classes, see . For more information about implementing .NET data providers, see .

*Depth*

*RemoveAt*

[C#] *int Depth {get;}*

[C++] *int get\_Depth();*

[VB] *ReadOnly Property Depth As Integer*

[JScript] *abstract function get Depth() : int;*

*Description*

*Gets a value indicating the depth of nesting for the current row.*

*The outermost table has a depth of zero.*

*IsClosed*

*RemoveAt*

[C#] *bool IsClosed {get;}*

[C++] *bool get\_IsClosed();*

[VB] *ReadOnly Property IsClosed As Boolean*

[JScript] *abstract function get IsClosed() : Boolean;*

*Description*

*Gets a value indicating whether the data reader is closed.*

***System.Data.IDataReader.IsClosed and***

***System.Data.IDataReader.RecordsAffected are the only properties that you can call after the System.Data.IDataReader is closed.***

***RecordsAffected***

***RemoveAt***

***[C#] int RecordsAffected {get;}***

***[C++] int get\_RecordsAffected();***

***[VB] ReadOnly Property RecordsAffected As Integer***

***[JScript] abstract function get RecordsAffected() : int;***

***Description***

***Gets the number of rows changed, inserted, or deleted by execution of the SQL statement.***

***The System.Data.IDataReader.RecordsAffected property is not set until all rows are read and you close the System.Data.IDataReader .***

***Close***

***[C#] void Close();***

***[C++] void Close();***

***[VB] Sub Close()***

***[JScript] function Close();***

***Description***

***Closes the System.Data.IDataReader Object.***

1       You must explicitly call the ***System.Data.IDataReader.Close*** method when  
2       you are through using the ***System.Data.IDataReader*** to use the associated  
3       ***System.Data.IDbConnection*** for any other purpose.

#### 4       *GetSchemaTable*

5  
6       [C#] *DataTable GetSchemaTable();*  
7       [C++] *DataTable\* GetSchemaTable();*  
8       [VB] *Function GetSchemaTable() As DataTable*  
9       [JScript] *function GetSchemaTable() : DataTable;*

#### 11      *Description*

12       Returns a ***System.Data.DataTable*** that describes the column metadata of  
13       the ***System.Data.IDataReader*** .

14       Return Value: A ***System.Data.DataTable*** that describes the column metadata.

15       The implementation of ***System.Data.IDataReader.GetSchemaTable***  
16       method for the OLE DB .NET Data Provider maps to the OLE DB  
17       ***IColumnRowset::GetColumnsRowset*** method, while the implementation for the  
18       SQL Server .NET Data Provider does not use an OLE DB provider layer.

#### 19      *NextResult*

20  
21      [C#] *bool NextResult();*  
22      [C++] *bool NextResult();*  
23      [VB] *Function NextResult() As Boolean*  
24      [JScript] *function NextResult() : Boolean;*

## Description

Advances the data reader to the next result, when reading the results of batch SQL statements.

Return Value: **true** if there are more rows; otherwise, **false**.

Used to process multiple results, which can be obtained by executing batch SQL statements.

## Read

[C#] **bool Read();**

[C++] **bool Read();**

[VB] **Function Read() As Boolean**

[JScript] **function Read() : Boolean;**

## Description

Advances the **System.Data.IDataReader** to the next record.

Return Value: **true** if there are more rows; otherwise, **false**.

The default position of the **System.Data.IDataReader** is prior to the first record. Therefore you must call **System.Data.IDataReader.Read** to begin accessing any data.

**IDataRecord** interface (**System.Data**)

## Read

## Description

Provides access to the column values within each row for a *DataReader*, and is implemented by .NET data providers that access relational databases.

The ***System.Data.IDataReader*** and ***System.Data.IDataRecord*** interfaces allow an inheriting class to implement a *DataReader* class, which provides a means of reading one or more forward-only streams of result sets. For more information about *DataReader* classes, see . For more information about implementing .NET data providers, see .

*FieldCount*

*Read*

[C#] *int FieldCount {get;}*

[C++] *int get\_FieldCount();*

[VB] *ReadOnly Property FieldCount As Integer*

[JScript] *abstract function get FieldCount() : int;*

*Description*

*Gets the number of columns in the current row.*

*After executing a query that does not return rows (for example, using the ***System.Data.IDbCommand.ExecuteNonQuery*** method), ***System.Data.IDataRecord.FieldCount*** returns -1.*

*Item*

*Read*

[C#] *object this[string name] {get;}*

[C++] *Object\* get\_Item(String\* name);*

1 *[VB] Default ReadOnly Property Item(ByVal name As String) As Object*

2 *[JScript] abstract returnValue = IDataRecordObject.Item(name);*

3

4 *Description*

5 *Gets the column with the specified name. The name of the column to find.*

6 *Item*

7 *Read*

8

9 *[C#] object this[int i] {get;}*

10 *[C++] Object\* get\_Item(int i);*

11 *[VB] Default ReadOnly Property Item(ByVal i As Integer) As Object*

12 *[JScript] abstract returnValue = IDataRecordObject.Item(i); Gets the specified*  
13 *column.*

14

15 *Description*

16 *Gets the column located at the specified index. The index of the column to*  
17 *get.*

18 *GetBoolean*

19

20 *[C#] bool GetBoolean(int i);*

21 *[C++] bool GetBoolean(int i);*

22 *[VB] Function GetBoolean(ByVal i As Integer) As Boolean*

23 *[JScript] function GetBoolean(i : int) : Boolean;*

24

25 *Description*

*Gets the boolean value of the specified column.*

*Return Value:* The value of the column.

*No conversions are performed, therefore the data retrieved must already be a boolean or an exception is generated. The index of the field to find.*

#### *GetByte*

*[C#] byte GetByte(int i);*

*[C++] unsigned char GetByte(int i);*

*[VB] Function GetByte(ByVal i As Integer) As Byte*

*[JScript] function GetByte(i : int) : Byte;*

#### *Description*

*Gets the 8-bit unsigned integer value of the specified field.*

*Return Value:* The 8-bit unsigned integer value of the specified field. The index of the field to find.

#### *GetBytes*

*[C#] long GetBytes(int i, long fieldOffset, byte[] buffer, int bufferoffset, int length);*

*[C++] \_\_int64 GetBytes(int i, \_\_int64 fieldOffset, unsigned char buffer \_\_gc[], int bufferoffset, int length);*

*[VB] Function GetBytes(ByVal i As Integer, ByVal fieldOffset As Long, ByVal buffer() As Byte, ByVal bufferoffset As Integer, ByVal length As Integer) As Long*

*[JScript] function GetBytes(i : int, fieldOffset : long, buffer : Byte[], bufferoffset : int, length : int) : long;*

## *Description*

*Reads a stream of bytes from the field offset in the specified field into the buffer starting at the given buffer offset.*

*Return Value:* The actual number of bytes read.

*The actual number of bytes read can be less than the requested length, if the end of the row is reached. If you pass a buffer that is **null**, **System.Data.IDataRecord.GetBytes(System.Int32, System.Int64, System.Byte[], System.Int32, System.Int32)** returns the length of the row in bytes. The zero-based column ordinal. The index within the field from which to begin the read operation. The buffer into which to read the stream of bytes. The index for buffer to begin the read operation. The number of bytes to read.*

## *GetChar*

*[C#] char GetChar(int i);*

*[C++] \_\_wchar\_t GetChar(int i);*

*[VB] Function GetChar(ByVal i As Integer) As Char*

*[JScript] function GetChar(i : int) : Char;*

## *Description*

*Gets the character value of the specified field.*

*Return Value:* The character value of the specified field. The index of the field to find.

## *GetChars*

```

1
2 [C#] long GetChars(int i, long fieldoffset, char[] buffer, int bufferoffset, int
3 length);
4 [C++] __int64 GetChars(int i, __int64 fieldoffset, __wchar_t buffer __gc[], int
5 bufferoffset, int length);
6 [VB] Function GetChars(ByVal i As Integer, ByVal fieldoffset As Long, ByVal
7 buffer() As Char, ByVal bufferoffset As Integer, ByVal length As Integer) As Long
8 [JScript] function GetChars(i : int, fieldoffset : long, buffer : Char[], bufferoffset :
9 int, length : int) : long;
10

```

## Description

*Reads a stream of characters from the field offset in the specified field into the buffer starting at the given buffer offset.*

*Return Value: The actual number of characters read.*

*The actual number of characters read can be less than the requested length, if the end of the field is reached. If you pass a buffer that is **null**, **System.Data.IDataRecord.GetChars(System.Int32, System.Int64, System.Char[], System.Int32, System.Int32)** returns the length of the field in characters. The zero-based column ordinal. The index within the row from which to begin the read operation. The buffer into which to read the stream of bytes. The index for buffer to begin the read operation. The number of bytes to read.*

## GetData

```

24 [C#] IDataReader GetData(int i);
25 [C++] IDataReader* GetData(int i);

```



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] DateTime GetDateTime(int i);
[C++] DateTime GetDateTime(int i);
[VB] Function GetDateTime(ByVal i As Integer) As DateTime
[JScript] function GetDateTime(i : int) : DateTime;
```

*Description*

*Gets the date and time data value of the specified field.*

*Return Value: The date and time data value of the specified field. The index of the field to find.*

*GetDecimal*

```
[C#] decimal GetDecimal(int i);
[C++] Decimal GetDecimal(int i);
[VB] Function GetDecimal(ByVal i As Integer) As Decimal
[JScript] function GetDecimal(i : int) : Decimal;
```

*Description*

*Gets the fixed-position numeric value of the specified field.*

*Return Value: The fixed-position numeric value of the specified field. The index of the field to find.*

*GetDouble*

```
[C#] double GetDouble(int i);
[C++] double GetDouble(int i);
```

[VB] Function GetDouble(ByVal i As Integer) As Double

[JScript] function GetDouble(i : int) : double;

#### Description

*Gets the double-precision floating point number of the specified field.*

*Return Value: The double-precision floating point number of the specified field.*

*The index of the field to find.*

#### GetFieldType

[C#] Type GetFieldType(int i);

[C++] Type\* GetFieldType(int i);

[VB] Function GetFieldType(ByVal i As Integer) As Type

[JScript] function GetFieldType(i : int) : Type;

#### Description

*Gets the **System.Type** information corresponding to the type of*

***System.Object** that would be returned from*

***System.Data.IDataRecord.GetValue(System.Int32)** .*

*Return Value: The **System.Type** information corresponding to the type of*

***System.Object** that would be returned from*

***System.Data.IDataRecord.GetValue(System.Int32)** .*

*This information can be used to increase performance by indicating the strongly-typed accessor to call. (e.g. using GetInt32 is roughly ten times faster than using GetValue.) Returns the **System.Type** information corresponding to the*

1 *type of **System.Object** that would be returned from*

2 ***System.Data.IDataRecord.GetValue(System.Int32)** . The index of the field to find.*

3 ***GetFloat***

4  
5 *[C#] float **GetFloat**(int i);*

6 *[C++] float **GetFloat**(int i);*

7 *[VB] Function **GetFloat**(ByVal i As Integer) As Single*

8 *[JScript] function **GetFloat**(i : int) : float;*

9  
10 ***Description***

11 *Gets the single-precision floating point number of the specified field.*

12 ***Return Value:** The single-precision floating point number of the specified field.*

13 *The index of the field to find.*

14 ***GetGuid***

15  
16 *[C#] Guid **GetGuid**(int i);*

17 *[C++] Guid **GetGuid**(int i);*

18 *[VB] Function **GetGuid**(ByVal i As Integer) As Guid*

19 *[JScript] function **GetGuid**(i : int) : Guid;*

20  
21 ***Description***

22 *Returns the guid value of the specified field.*

23 ***Return Value:** The guid value of the specified field. The index of the field to find.*

24 ***GetInt16***

25

*[C#] short GetInt16(int i);*

*[C++] short GetInt16(int i);*

*[VB] Function GetInt16(ByVal i As Integer) As Short*

*[JScript] function GetInt16(i : int) : Int16;*

### *Description*

*Gets the 16-bit signed integer value of the specified field.*

*Return Value: The 16-bit signed integer value of the specified field. The index of the field to find.*

### *GetInt32*

*[C#] int GetInt32(int i);*

*[C++] int GetInt32(int i);*

*[VB] Function GetInt32(ByVal i As Integer) As Integer*

*[JScript] function GetInt32(i : int) : int;*

### *Description*

*Gets the 32-bit signed integer value of the specified field.*

*Return Value: The 32-bit signed integer value of the specified field. The index of the field to find.*

### *GetInt64*

*[C#] long GetInt64(int i);*

*[C++] \_\_int64 GetInt64(int i);*

*[VB] Function GetInt64(ByVal i As Integer) As Long*

*[JScript] function GetInt64(i : int) : long;*

#### *Description*

*Gets the 64-bit signed integer value of the specified field.*

*Return Value: The 64-bit signed integer value of the specified field. The index of the field to find.*

#### *GetName*

*[C#] string GetName(int i);*

*[C++] String\* GetName(int i);*

*[VB] Function GetName(ByVal i As Integer) As String*

*[JScript] function GetName(i : int) : String;*

#### *Description*

*Gets the name for the field to find.*

*Return Value: The name of the field or the empty string (""), if there is no value to return. The index of the field to find.*

#### *GetOrdinal*

*[C#] int GetOrdinal(string name);*

*[C++] int GetOrdinal(String\* name);*

*[VB] Function GetOrdinal(ByVal name As String) As Integer*

*[JScript] function GetOrdinal(name : String) : int;*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

*Return the index of the named field.*  
*Return Value: The index of the named field. The name of the field to find.*

```
GetString

[C#] string GetString(int i);
[C++] String* GetString(int i);
[VB] Function GetString(ByVal i As Integer) As String
[JScript] function GetString(i : int) : String;
```

*Description*

*Gets the string value of the specified field.*  
*Return Value: The string value of the specified field. The index of the field to find.*

```
GetValue

[C#] object GetValue(int i);
[C++] Object* GetValue(int i);
[VB] Function GetValue(ByVal i As Integer) As Object
[JScript] function GetValue(i : int) : Object;
```

*Description*

*Return the value of the specified field.*  
*Return Value: The **System.Object** which will contain the field value upon return.*  
*The index of the field to find.*

## GetValues

```
[C#] int GetValues(object[] values);
[C++] int GetValues(Object* values __gc[]);
[VB] Function GetValues(ByVal values() As Object) As Integer
[JScript] function GetValues(values : Object[]) : int;
```

### Description

*Gets all the attribute fields in the collection for the current record.*

*Return Value: The number of instances of **System.Object** in the array.*

*For most applications, the*

***System.Data.IDataRecord.GetValues(System.Object[])** method provides an efficient means for retrieving all columns, rather than retrieving each column individually. An array of **System.Object** to copy the attribute fields into.*

### IsDBNull

```
[C#] bool IsDBNull(int i);
[C++] bool IsDBNull(int i);
[VB] Function IsDBNull(ByVal i As Integer) As Boolean
[JScript] function IsDBNull(i : int) : Boolean;
```

### Description

*Return whether the specified field is set to null.*

*Return Value: **true** if the specified field is set to null, otherwise **false** . The index of the field to find.*

*IDbCommand interface (System.Data)*

*IsDBNull*

*Description*

*Represents a SQL statement that is executed while connected to a data source, and is implemented by .NET data providers that access relational databases.*

*The **System.Data.IDbCommand** interface allows an inheriting class to implement a Command class, which represents a SQL statement that is executed at a data source. For more information about Command classes, see . For more information about implementing .NET data providers, see .*

*CommandText*

*IsDBNull*

*[C#] string CommandText {get; set;}*

*[C++] String\* get\_CommandText();void set\_CommandText(String\*);*

*[VB] Property CommandText As String*

*[JScript] abstract function get CommandText() : String;public abstract function set CommandText(String);*

*Description*

*Gets or sets the text command to run against the data source.*

*When the **System.Data.IDbCommand.CommandType** property is set to **StoredProcedure** , set the **System.Data.IDbCommand.CommandText** property to*

the name of the stored procedure. The command will call this stored procedure when you call one of the Execute methods.

*CommandTimeout*

*IsDBNull*

[C#] *int CommandTimeout {get; set;}*

[C++] *int get\_CommandTimeout();void set\_CommandTimeout(int);*

[VB] *Property CommandTimeout As Integer*

[JScript] *abstract function get CommandTimeout() : int;public abstract function set CommandTimeout(int);*

#### *Description*

*Gets or sets the wait time before terminating the attempt to execute a command and generating an error.*

*CommandType*

*IsDBNull*

[C#] *CommandType CommandType {get; set;}*

[C++] *CommandType get\_CommandType();void set\_CommandType(CommandType);*

[VB] *Property CommandType As CommandType*

[JScript] *abstract function get CommandType() : CommandType;public abstract function set CommandType(CommandType);*

#### *Description*

Indicates or specifies how the **System.Data.IDbCommand.CommandText** property is interpreted.

When you set the **System.Data.IDbCommand.CommandType** property to **StoredProcedure**, you should set the **System.Data.IDbCommand.CommandText** property to the name of the stored procedure. The command executes this stored procedure when you call one of the **Execute** methods.

**Connection**

**IsDBNull**

[C#] **IDbConnection Connection {get; set;}**

[C++] **IDbConnection\* get\_Connection(); void set\_Connection(IDbConnection\*);**

[VB] **Property Connection As IDbConnection**

[JScript] **abstract function get Connection() : IDbConnection; public abstract function set Connection(IDbConnection);**

**Description**

**Gets or sets the System.Data.IDbConnection used by this instance of the System.Data.IDbCommand.**

**Parameters**

**IsDBNull**

[C#] **IDataParameterCollection Parameters {get;}**

[C++] **IDataParameterCollection\* get\_Parameters();**

[VB] **ReadOnly Property Parameters As IDataParameterCollection**

[JScript] **abstract function get Parameters() : IDataParameterCollection;**

*Description*

*Gets the **System.Data.IDataParameterCollection** .*

*Transaction*

*IsDBNull*

*[C#] IDbTransaction Transaction {get; set;}*

*[C++] IDbTransaction\* get\_Transaction();void*

*set\_Transaction(IDbTransaction\*);*

*[VB] Property Transaction As IDbTransaction*

*[JScript] abstract function get Transaction() : IDbTransaction;public abstract*

*function set Transaction(IDbTransaction);*

*Description*

*Gets or sets the transaction in which the **Command** object of an ADO .NET data provider executes.*

*UpdatedRowSource*

*IsDBNull*

*[C#] UpdateRowSource UpdatedRowSource {get; set;}*

*[C++] UpdateRowSource get\_UpdatedRowSource();void*

*set\_UpdatedRowSource(UpdateRowSource);*

*[VB] Property UpdatedRowSource As UpdateRowSource*

*[JScript] abstract function get UpdatedRowSource() : UpdateRowSource;public*

*abstract function set UpdatedRowSource(UpdateRowSource);*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

*Gets or sets how command results are applied to the **System.Data.DataRow** when used by the **System.Data.IDataAdapter.Update(System.Data.DataSet)** method of a **System.Data.Common.DbDataAdapter** .*

*Cancel*

*[C#] void Cancel();  
[C++] void Cancel();  
[VB] Sub Cancel()  
[JScript] function Cancel();*

*Description*

*Cancels the execution of an **System.Data.IDbCommand** .*

*CreateParameter*

*[C#] IDbDataParameter CreateParameter();  
[C++] IDbDataParameter\* CreateParameter();  
[VB] Function CreateParameter() As IDbDataParameter  
[JScript] function CreateParameter() : IDbDataParameter;*

*Description*

*Creates a new instance of an ADO .NET **Parameter** object.*

*Return Value: An ADO .NET **Parameter** object.*

When inheriting from **System.Data.IDbCommand**, an ADO .NET data provider implements a strongly-typed version of **System.Data.IDbCommand.CreateParameter**.

### *ExecuteNonQuery*

```
[C#] int ExecuteNonQuery();
[C++] int ExecuteNonQuery();
[VB] Function ExecuteNonQuery() As Integer
[JScript] function ExecuteNonQuery() : int;
```

### *Description*

Executes a SQL statement against the **Connection** object of an ADO .NET data provider, and returns the number of rows affected.

**Return Value:** The number of rows affected.

You can use the **System.Data.IDbCommand.ExecuteNonQuery** to perform catalog operations (for example, querying the structure of a database or creating database objects such as tables), or to change the data in a database without using a **System.Data.DataSet** by executing UPDATE, INSERT, or DELETE statements.

### *ExecuteReader*

```
[C#] IDataReader ExecuteReader();
[C++] IDataReader* ExecuteReader();
[VB] Function ExecuteReader() As IDataReader
[JScript] function ExecuteReader() : IDataReader; Executes the
```

*System.Data.IDbCommand.CommandText* against the  
*System.Data.IDbCommand.Connection* and builds an *System.Data.IDataReader*

#### *Description*

*Executes the System.Data.IDbCommand.CommandText* against the  
*System.Data.IDbCommand.Connection* and builds an *System.Data.IDataReader*

*Return Value:* An *System.Data.IDataReader* object.

#### *ExecuteReader*

*[C#] IDataReader ExecuteReader(CommandBehavior behavior);*

*[C++] IDataReader\* ExecuteReader(CommandBehavior behavior);*

*[VB] Function ExecuteReader(ByVal behavior As CommandBehavior) As  
 IDataReader*

*[JScript] function ExecuteReader(behavior : CommandBehavior) : IDataReader;*

#### *Description*

*Executes the System.Data.IDbCommand.CommandText* against the  
*System.Data.IDbCommand.Connection* , and builds an  
*System.Data.IDataReader* using one of the *System.Data.CommandBehavior*  
 values.

*Return Value:* An *System.Data.IDataReader* object.

The caller must call the **System.Data.IDbConnection.Open** method of the **System.Data.IDbCommand.Connection** property. One of the **System.Data.CommandBehavior** values.

### *ExecuteScalar*

[C#] *object ExecuteScalar();*

[C++] *Object\* ExecuteScalar();*

[VB] *Function ExecuteScalar() As Object*

[JScript] *function ExecuteScalar() : Object;*

### *Description*

Executes the query, and returns the first column of the first row in the resultset returned by the query. Extra columns or rows are ignored.

**Return Value:** The first column of the first row in the resultset.

Use the **System.Data.IDbCommand.ExecuteScalar** method to retrieve a single value (for example, an aggregate value) from a database. This requires less code than using the **System.Data.IDbCommand.ExecuteReader** method, and then performing the operations necessary to generate the single value using the data returned by an **System.Data.IDbDataReader**.

### *Prepare*

[C#] *void Prepare();*

[C++] *void Prepare();*

[VB] *Sub Prepare()*

[JScript] *function Prepare();*

## Description

Creates a prepared (or compiled) version of the command on the data source.

If the **System.Data.IDbCommand.CommandType** property is set to **TableDirect**, **System.Data.IDbCommand.Prepare** does nothing. If **System.Data.IDbCommand.CommandType** is set to **StoredProcedure**, the call to **System.Data.IDbCommand.Prepare** should succeed, although it may result in a no-op.

*IDbConnection interface (System.Data)*

*Prepare*

## Description

Represents an open connection to a data source, and is implemented by .NET data providers that access relational databases.

The **System.Data.IDbConnection** interface allows an inheriting class to implement a *Connection* class, which represents a unique session with a data source (for example, a network connection to a server). For more information about *Connection* classes, see . For more information about implementing .NET data providers, see .

*ConnectionString*

*Prepare*

[C#] string ConnectionString {get; set;}

1 *[C++] String\* get\_ConnectionString();void set\_ConnectionString(String\*);*  
 2 *[VB] Property ConnectionString As String*  
 3 *[JScript] abstract function get ConnectionString() : String;public abstract*  
 4 *function set ConnectionString(String);*

5  
 6 *Description*

7 *Gets or sets the string used to open a database.*  
 8 *The string can only be set when the connection state is closed.*

9 *ConnectionTimeout*

10 *Prepare*

11  
 12 *[C#] int ConnectionTimeout {get;}*

13 *[C++] int get\_ConnectionTimeout();*

14 *[VB] ReadOnly Property ConnectionTimeout As Integer*

15 *[JScript] abstract function get ConnectionTimeout() : int;*

16  
 17 *Description*

18 *Gets the time to wait while trying to establish a connection before*  
 19 *terminating the attempt and generating an error.*

20 *A value of 0 indicates no limit, and should be avoided in a*  
 21 ***System.Data.IDbConnection.ConnectionString** because an attempt to connect*  
 22 *will wait indefinitely.*

23 *Database*

24 *Prepare*

25

1  
2 *[C#] string Database {get;}*

3 *[C++] String\* get\_Database();*

4 *[VB] ReadOnly Property Database As String*

5 *[JScript] abstract function get Database() : String;*

6  
7 *Description*

8 *Gets the name of the current database or the database to be used once a*  
9 *connection is open.*

10 *The **System.Data.OleDb.OleDbConnection.Database** property updates*  
11 *dynamically. If you change the current database using a SQL statement or the*  
12 ***System.Data.OleDb.OleDbConnection.ChangeDatabase(System.String)** method,*  
13 *an informational message is sent and the property is updated automatically.*

14 *State*

15 *Prepare*

16  
17 *[C#] ConnectionState State {get;}*

18 *[C++] ConnectionState get\_State();*

19 *[VB] ReadOnly Property State As ConnectionState*

20 *[JScript] abstract function get State() : ConnectionState;*

21  
22 *Description*

23 *Gets the current state of the connection.*

24 ***System.Data.ConnectionState** values may be OR'ed together.*

25 *BeginTransaction*

*[C#] IDbTransaction BeginTransaction();*

*[C++] IDbTransaction\* BeginTransaction();*

*[VB] Function BeginTransaction() As IDbTransaction*

*[JScript] function BeginTransaction() : IDbTransaction; Begins a database transaction.*

### *Description*

*Begins a database transaction.*

*You must explicitly commit or roll back the transaction using the **System.Data.IDbTransaction.Commit** or **System.Data.IDbTransaction.Rollback** method. To ensure that the SQL Server .NET Data Provider transaction management model performs correctly, avoid using other transaction management models, such as the one provided by SQL Server.*

### *BeginTransaction*

*[C#] IDbTransaction BeginTransaction(IsolationLevel il);*

*[C++] IDbTransaction\* BeginTransaction(IsolationLevel il);*

*[VB] Function BeginTransaction(ByVal il As IsolationLevel) As IDbTransaction*

*[JScript] function BeginTransaction(il : IsolationLevel) : IDbTransaction;*

### *Description*

*Begins a database transaction with the specified isolation level.*

*Return Value: An object representing the new transaction.*

*You must explicitly commit or roll back the transaction using the **System.Data.IDbTransaction.Commit** or **System.Data.IDbTransaction.Rollback** method. To ensure that the SQL Server .NET Data Provider transaction management model performs correctly, avoid using other transaction management models, such as the one provided by SQL Server. The isolation level under which the transaction should run.*

### *ChangeDatabase*

*[C#] void ChangeDatabase(string databaseName);*  
*[C++] void ChangeDatabase(String\* databaseName);*  
*[VB] Sub ChangeDatabase(ByVal databaseName As String)*  
*[JScript] function ChangeDatabase(databaseName : String);*

### *Description*

*Changes the current database for an open **System.Data.IDbConnection**. The database name.*

### *Close*

*[C#] void Close();*  
*[C++] void Close();*  
*[VB] Sub Close()*  
*[JScript] function Close();*

### *Description*

*Closes the connection to the database.*

*The **System.Data.SqlClient.SqlConnection.Close** method rolls back any pending transactions. It then releases the connection to the connection pool, or closes the connection if connection pooling is disabled.*

#### *CreateCommand*

*[C#] IDbCommand CreateCommand();*  
*[C++] IDbCommand\* CreateCommand();*  
*[VB] Function CreateCommand() As IDbCommand*  
*[JScript] function CreateCommand() : IDbCommand;*

#### *Description*

*Creates and returns an **System.Data.IDbCommand** object associated with the **System.Data.IDbConnection** .*

#### *Open*

*[C#] void Open();*  
*[C++] void Open();*  
*[VB] Sub Open()*  
*[JScript] function Open();*

#### *Description*

*Opens a database connection with the property settings specified by the **System.Data.IDbConnection.ConnectionString** .*

*When overriding **System.Data.OleDb.OleDbConnection.Open** in a derived class, opens a connection to the data source.*

*IDbDataAdapter interface (System.Data)*

*Open*

*Description*

*Represents a set of command-related properties that are used to fill the **System.Data.DataSet** and update a data source, and is implemented by .NET data providers that access relational databases.*

*The **System.Data.IDbDataAdapter** interface inherits from the **System.Data.IDataAdapter** interface and allows an object to create a **DataAdapter** designed for use with a relational database. The **System.Data.IDbDataAdapter** interface and, optionally, the utility class, **System.Data.Common.DbDataAdapter**, allow an inheriting class to implement a **DataAdapter** class, which represents the bridge between a data source and a **System.Data.DataSet**. For more information about **DataAdapter** classes, see . For more information about implementing .NET data providers, see .*

*DeleteCommand*

*Open*

*[C#] IDbCommand DeleteCommand {get; set;}*

*[C++] IDbCommand\* get\_DeleteCommand();void*

*set\_DeleteCommand(IDbCommand\*);*

*[VB] Property DeleteCommand As IDbCommand*

*[JScript] abstract function get DeleteCommand() : IDbCommand;public abstract*

*function set DeleteCommand(IDbCommand);*

## *Description*

*Gets or sets an SQL statement for deleting records from the data set.*

## *During*

*System.Data.Common.DbDataAdapter.Update(System.Data.DataSet) , if this property is not set and primary key information is present in the System.Data.DataSet , the System.Data.IDbDataAdapter.DeleteCommand can be generated automatically if you set the SelectCommand property of a .NET data provider. Then, any additional SQL statements that you do not set are generated by the CommandBuilder. This generation logic requires key column information to be present in the System.Data.DataSet . For more information see .*

## *InsertCommand*

## *Open*

*[C#] IDbCommand InsertCommand {get; set;}*

*[C++] IDbCommand\* get\_InsertCommand();void*

*set\_InsertCommand(IDbCommand\*);*

*[VB] Property InsertCommand As IDbCommand*

*[JScript] abstract function get InsertCommand() : IDbCommand;public abstract function set InsertCommand(IDbCommand);*

## *Description*

*Gets or sets an SQL statement used to insert new records into the data source.*

*During*

*System.Data.Common.DbDataAdapter.Update(System.Data.DataSet) , if this property is not set and primary key information is present in the System.Data.DataSet , the System.Data.IDbDataAdapter.InsertCommand can be generated automatically if you set the SelectCommand property of a .NET data provider. Then, any additional SQL statements that you do not set are generated by the CommandBuilder. This generation logic requires key column information to be present in the System.Data.DataSet . For more information see .*

*SelectCommand*

*Open*

```
[C#] IDbCommand SelectCommand {get; set;}
[C++] IDbCommand* get_SelectCommand();void
set_SelectCommand(IDbCommand*);
[VB] Property SelectCommand As IDbCommand
[JScript] abstract function get SelectCommand() : IDbCommand;public abstract
function set SelectCommand(IDbCommand);
```

*Description*

*Gets or sets an SQL statement used to select records in the data source.*

*When System.Data.IDbDataAdapter.SelectCommand is assigned to a previously created System.Data.IDbCommand , the System.Data.IDbCommand is not cloned. The System.Data.IDbDataAdapter.SelectCommand maintains a reference to the previously created System.Data.IDbCommand object.*

*UpdateCommand*

## Open

[C#] IDbCommand UpdateCommand {get; set;}

[C++] IDbCommand\* get\_UpdateCommand();void

set\_UpdateCommand(IDbCommand\*);

[VB] Property UpdateCommand As IDbCommand

[JScript] abstract function get UpdateCommand() : IDbCommand;public abstract

function set UpdateCommand(IDbCommand);

## Description

*Gets or sets an SQL statement used to update records in the data source.*

## During

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** , if this

property is not set and primary key information is present in the

**System.Data.DataSet** , the **System.Data.IDbDataAdapter.UpdateCommand** can

be generated automatically if you set the **SelectCommand** property of a .NET data

provider. Then, any additional SQL statements that you do not set are generated

by the **CommandBuilder**. This generation logic requires key column information to

be present in the **System.Data.DataSet** . For more information see .

*IDbDataParameter interface (System.Data)*

## Open

## Precision

## Open

## Scale

## Open

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Size*

*Open*

*IDbTransaction interface (System.Data)*

*Open*

*Description*

*Represents a transaction to be performed at a data source, and is implemented by .NET data providers that access relational databases.*

*The **System.Data.IDbTransaction** interface allows an inheriting class to implement a Transaction class, which represents the transaction to be performed at a data source. For more information about Transaction classes, see . For more information about implementing .NET data providers, see .*

*Connection*

*Open*

*[C#] IDbConnection Connection {get;}*

*[C++] IDbConnection\* get\_Connection();*

*[VB] ReadOnly Property Connection As IDbConnection*

*[JScript] abstract function get Connection() : IDbConnection;*

*Description*

*IsolationLevel*

*Open*

1  
2 *[C#] IsolationLevel IsolationLevel {get;}*  
3 *[C++] IsolationLevel get\_IsolationLevel();*  
4 *[VB] ReadOnly Property IsolationLevel As IsolationLevel*  
5 *[JScript] abstract function get IsolationLevel() : IsolationLevel;*

6  
7 *Description*

8 *Specifies the **System.Data.IsolationLevel** for this transaction.*  
9 *Parallel transactions are not supported. Therefore, the*  
10 ***System.Data.IsolationLevel** applies to the entire transaction.*

11 *Commit*

12  
13 *[C#] void Commit();*  
14 *[C++] void Commit();*  
15 *[VB] Sub Commit()*  
16 *[JScript] function Commit();*

17  
18 *Description*

19 *Commits the database transaction.*

20 *Rollback*

21  
22 *[C#] void Rollback();*  
23 *[C++] void Rollback();*  
24 *[VB] Sub Rollback()*  
25 *[JScript] function Rollback();*

*Description*

*Rolls back a transaction from a pending state.*

*The transaction can only be rolled back from a pending state (after **System.Data.IDbConnection.BeginTransaction** has been called, but before **System.Data.IDbTransaction.Commit** is called).*

*InRowChangingEventException class (System.Data)*

*Rollback*

*Description*

*Represents the exception that is thrown when when calling the **System.Data.DataRow.EndEdit** method within the **System.Data.DataTable.RowChanging** event.*

*InRowChangingEventException*

*Example Syntax:*

*Rollback*

*[C#] public InRowChangingEventException();*

*[C++] public: InRowChangingEventException();*

*[VB] Public Sub New()*

*[JScript] public function InRowChangingEventException();*

*Description*

1        *Initializes a new instance of the*  
 2        ***System.Data.InRowChangingEventException*** class.

3        *InRowChangingEventException*

4        *Example Syntax:*

5        *Rollback*

6  
 7        [C#] *public InRowChangingEventException(string s);*

8        [C++] *public: InRowChangingEventException(String\* s);*

9        [VB] *Public Sub New(ByVal s As String)*

10        [JScript] *public function InRowChangingEventException(s : String);*

11  
 12        *Description*

13        *Initializes a new instance of the*  
 14        ***System.Data.InRowChangingEventException*** class with the specified string. The  
 15        string to display when the exception is thrown.

16        *InRowChangingEventException*

17        *Example Syntax:*

18        *Rollback*

19  
 20        [C#] *public InRowChangingEventException(SerializationInfo info,*  
 21        *StreamingContext context);*

22        [C++] *public: InRowChangingEventException(SerializationInfo\* info,*  
 23        *StreamingContext context);*

24        [VB] *Public Sub New(ByVal info As SerializationInfo, ByVal context As*  
 25        *StreamingContext)*

1 *[JScript] public function InRowChangingEventException(info : SerializationInfo,*  
 2 *context : StreamingContext); Initializes a new instance of the*  
 3 ***System.Data.InRowChangingEventException** class.*

4  
 5 *Description*

6 *Initializes a new instance of the*  
 7 ***System.Data.InRowChangingEventException** class with serialization*  
 8 *information. The data necessary to serialize or deserialize an object. Description*  
 9 *of the source and destination of the specified serialized stream.*

10 *HelpLink*

11 *HResult*

12 *InnerException*

13 *Message*

14 *Source*

15 *StackTrace*

16 *TargetSite*

17 *InternalDataCollectionBase class (System.Data)*

18 *ToString*

19  
 20  
 21 *Description*

22 *Provides the base functionality for creating collections.*

23 *The **System.BaseCollection** class and its members are not intended for use*  
 24 *as a stand alone component, but as the class from which other collection classes*  
 25 *derive standard functionality.*

*InternalDataCollectionBase*

*Example Syntax:*

*ToString*

[C#] *public InternalDataCollectionBase();*

[C++] *public: InternalDataCollectionBase();*

[VB] *Public Sub New()*

[JScript] *public function InternalDataCollectionBase();*

*Count*

*ToString*

[C#] *public virtual int Count {get;}*

[C++] *public: \_\_property virtual int get\_Count();*

[VB] *Overridable Public ReadOnly Property Count As Integer*

[JScript] *public function get Count() : int;*

*Description*

*Gets the total number of elements in a collection.*

*The **System.BaseCollection** class and its members are not intended for use as a stand alone component, but as the class from which other collection classes derive standard functionality.*

*IsReadOnly*

*ToString*

[C#] *public bool IsReadOnly {get;}*

1 [C++] public: \_\_property bool get\_IsReadOnly();

2 [VB] Public ReadOnly Property IsReadOnly As Boolean

3 [JScript] public function get IsReadOnly() : Boolean;

4  
5 *Description*

6 Gets a value indicating whether the  
7 **System.Data.InternalDataCollectionBase** is read-only.

8 The **System.BaseCollection** class and its members are not intended for use  
9 as a stand alone component, but as the class from which other collection classes  
10 derive standard functionality.

11 *IsSynchronized*

12 *ToString*

13  
14 [C#] public bool IsSynchronized {get;}

15 [C++] public: \_\_property bool get\_IsSynchronized();

16 [VB] Public ReadOnly Property IsSynchronized As Boolean

17 [JScript] public function get IsSynchronized() : Boolean;

18  
19 *Description*

20 Gets a value indicating whether the  
21 **System.Data.InternalDataCollectionBase** is synchronized.

22 The **System.BaseCollection** class and its members are not intended for use  
23 as a stand alone component, but as the class from which other collection classes  
24 derive standard functionality.

25 *List*

## *ToString*

*[C#] protected virtual ArrayList List {get;}*

*[C++] protected: \_\_property virtual ArrayList\* get\_List();*

*[VB] Overridable Protected ReadOnly Property List As ArrayList*

*[JScript] protected function get List() : ArrayList;*

## *Description*

*Gets the items of the collection as a list.*

*The **System.BaseCollection** class and its members are not intended for use as a stand alone component, but as the class from which other collection classes derive standard functionality.*

## *SyncRoot*

## *ToString*

*[C#] public object SyncRoot {get;}*

*[C++] public: \_\_property Object\* get\_SyncRoot();*

*[VB] Public ReadOnly Property SyncRoot As Object*

*[JScript] public function get SyncRoot() : Object;*

## *Description*

*Gets an object that can be used to synchronize the collection.*

*The **System.BaseCollection** class and its members are not intended for use as a stand alone component, but as the class from which other collection classes derive standard functionality.*

## *CopyTo*

*[C#] public void CopyTo(Array ar, int index);*

*[C++] public: \_\_sealed void CopyTo(Array\* ar, int index);*

*[VB] NotOverridable Public Sub CopyTo(ByVal ar As Array, ByVal index As Integer)*

*[JScript] public function CopyTo(ar : Array, index : int);*

### *Description*

*Copies all the elements of the current **System.Data.InternalDataCollectionBase** to a one-dimensional **System.Array**, starting at the specified **System.Data.InternalDataCollectionBase** index.*

*This method can be overridden by a derived class. The one-dimensional **System.Array** to copy the current **System.Data.InternalDataCollectionBase** object's elements into. The destination **System.Array** index to start copying into.*

### *GetEnumerator*

*[C#] public IEnumerator GetEnumerator();*

*[C++] public: \_\_sealed IEnumerator\* GetEnumerator();*

*[VB] NotOverridable Public Function GetEnumerator() As IEnumerator*

*[JScript] public function GetEnumerator() : IEnumerator;*

### *Description*

*Gets an **System.Collections.IEnumerator** for the collection.*

*Return Value: An **System.Collections.IEnumerator** for the collection.*

*The **System.BaseCollection** class and its members are not intended for use as a stand alone component, but as the class from which other collection classes derive standard functionality.*

***InvalidConstraintException** class (**System.Data**)*

***ToString***

#### *Description*

*Represents the exception that is thrown when incorrectly attempting to create or access a relation.*

*The **System.Data.InvalidConstraintException** is thrown when incorrectly invoking the following methods while attempting to create or access a relation.*

***InvalidConstraintException***

*Example Syntax:*

***ToString***

*[C#] public **InvalidConstraintException**();*

*[C++] public: **InvalidConstraintException**();*

*[VB] Public Sub New()*

*[JScript] public function **InvalidConstraintException**();*

#### *Description*

*Initializes a new instance of the **System.Data.InvalidConstraintException** class.*

***InvalidConstraintException***

*Example Syntax:*

*ToString*

*[C#] public InvalidConstraintException(string s);*

*[C++] public: InvalidConstraintException(String\* s);*

*[VB] Public Sub New(ByVal s As String)*

*[JScript] public function InvalidConstraintException(s : String);*

*Description*

*Initializes a new instance of the **System.Data.InvalidConstraintException** class with the specified string. The string to display when the exception is thrown.*

*InvalidConstraintException*

*Example Syntax:*

*ToString*

*[C#] public InvalidConstraintException(SerializationInfo info, StreamingContext context);*

*[C++] public: InvalidConstraintException(SerializationInfo\* info, StreamingContext context);*

*[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)*

*[JScript] public function InvalidConstraintException(info : SerializationInfo, context : StreamingContext);* *Initializes a new instance of the **System.Data.InvalidConstraintException** class.*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

*Initializes a new instance of the **System.Data.InvalidConstraintException** class with serialization information. The data necessary to serialize or deserialize an object. Description of the source and destination of the specified serialized stream.*

*HelpLink*

*HResult*

*InnerException*

*Message*

*Source*

*StackTrace*

*TargetSite*

*InvalidExpressionException class (System.Data)*

*ToString*

*Description*

*Represents the exception that is thrown when attempting to add a **System.Data.DataColumn** containing an invalid **System.Data.DataColumn.Expression** to a **System.Data.DataColumnCollection**. The **System.Data.DataColumn.Expression** property is use to calculate the value of a column, or create an aggregate column.*

*InvalidExpressionException*

*Example Syntax:*

1           *ToString*

2

3    [C#] *public InvalidExpressionException();*

4    [C++] *public: InvalidExpressionException();*

5    [VB] *Public Sub New()*

6    [JScript] *public function InvalidExpressionException();* *Initializes a new instance*

7    *of the System.Data.InvalidExpressionException class.*

8

9    *Description*

10       *Initializes a new instance of the System.Data.InvalidExpressionException*

11       *class.*

12       *InvalidExpressionException*

13       *Example Syntax:*

14       *ToString*

15

16    [C#] *public InvalidExpressionException(string s);*

17    [C++] *public: InvalidExpressionException(String\* s);*

18    [VB] *Public Sub New(ByVal s As String)*

19    [JScript] *public function InvalidExpressionException(s : String);*

20

21    *Description*

22       *Initializes a new instance of the System.Data.InvalidExpressionException*

23       *class with the specified string. The string to display when the exception is thrown.*

24       *InvalidExpressionException*

25       *Example Syntax:*

## *ToString*

*[C#] public InvalidExpressionException(SerializationInfo info, StreamingContext context);*

*[C++] public: InvalidExpressionException(SerializationInfo\* info, StreamingContext context);*

*[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)*

*[JScript] public function InvalidExpressionException(info : SerializationInfo, context : StreamingContext);*

## *Description*

*Initializes a new instance of the **System.Data.InvalidExpressionException** class with the **System.Runtime.Serialization.SerializationInfo** and the **System.Runtime.Serialization.StreamingContext** . The data needed to serialize or deserialize an object. The source and destination of a given serialized stream.*

*HelpLink*

*HResult*

*InnerException*

*Message*

*Source*

*StackTrace*

*TargetSite*

*IsolationLevel enumeration (System.Data)*

*ToString*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

*Specifies the transaction locking behavior for the connection.*

*The **System.Data.IsolationLevel** values are used by a .NET data provider when performing a transaction.*

*ToString*

*[C#] public const IsolationLevel Chaos;*

*[C++] public: const IsolationLevel Chaos;*

*[VB] Public Const Chaos As IsolationLevel*

*[JScript] public var Chaos : IsolationLevel;*

*Description*

*The pending changes from more highly isolated transactions cannot be overwritten.*

*ToString*

*[C#] public const IsolationLevel ReadCommitted;*

*[C++] public: const IsolationLevel ReadCommitted;*

*[VB] Public Const ReadCommitted As IsolationLevel*

*[JScript] public var ReadCommitted : IsolationLevel;*

*Description*

1        *Shared locks are held while the data is being read to avoid dirty reads, but*  
2 *the data can be changed before the end of the transaction, resulting in non-*  
3 *repeatable reads or phantom data.*

4        *ToString*

5  
6        *[C#] public const IsolationLevel ReadUncommitted;*  
7        *[C++] public: const IsolationLevel ReadUncommitted;*  
8        *[VB] Public Const ReadUncommitted As IsolationLevel*  
9        *[JScript] public var ReadUncommitted : IsolationLevel;*

10  
11        *Description*

12        *A dirty read is possible, meaning that no shared locks are issued and no*  
13 *exclusive locks are honored.*

14        *ToString*

15  
16        *[C#] public const IsolationLevel RepeatableRead;*  
17        *[C++] public: const IsolationLevel RepeatableRead;*  
18        *[VB] Public Const RepeatableRead As IsolationLevel*  
19        *[JScript] public var RepeatableRead : IsolationLevel;*

20  
21        *Description*

22        *Locks are placed on all data that is used in a query, preventing other users*  
23 *from updating the data. Prevents non-repeatable reads but phantom rows are still*  
24 *possible.*

25        *ToString*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const IsolationLevel Serializable;
[C++] public: const IsolationLevel Serializable;
[VB] Public Const Serializable As IsolationLevel
[JScript] public var Serializable : IsolationLevel;
```

*Description*

*A range lock is palced on the **System.Data.DataSet** , preventing other users from updating or inserting rows into the dataset until the transaction is complete.*

*ToString*

```
[C#] public const IsolationLevel Unspecified;
[C++] public: const IsolationLevel Unspecified;
[VB] Public Const Unspecified As IsolationLevel
[JScript] public var Unspecified : IsolationLevel;
```

*Description*

*A different isolation level than the one specified is being used, but the level cannot be determined.*

*ITableMapping interface (System.Data)*

*ToString*

*Description*

Associates a source table with a table in a **System.Data.DataSet** , and is implemented by the **System.Data.Common.DataTableMapping** class, which is used in common by .NET data providers.

The **System.Data.ITableMapping** interface allows an inheriting class to implement a **TableMapping** class, which associates a data source column with a **System.Data.DataSet** column. For more information, see .

**ColumnMappings**

**ToString**

[C#] **ICollection** **ColumnMappings** {get;}

[C++] **ICollection\*** **get\_ColumnMappings()**;

[VB] **ReadOnly Property** **ColumnMappings** **As ICollection**

[JScript] **abstract function** **get ColumnMappings()** : **ICollection**;

**Description**

Gets the derived **System.Data.Common.DataColumnMappingCollection** for the **System.Data.DataTable** .

**DataSetTable**

**ToString**

[C#] **string** **DataSetTable** {get; set;}

[C++] **String\*** **get\_DataSetTable()**; **void set\_DataSetTable(String\*)**;

[VB] **Property** **DataSetTable** **As String**

[JScript] **abstract function** **get DataSetTable()** : **String**; **public abstract function** **set DataSetTable(String)**;

1  
2 *Description*

3 *Gets or sets the case-insensitive name of the table within the*

4 ***System.Data.DataSet .***

5 *SourceTable*

6 *ToString*

7  
8 *[C#] string SourceTable {get; set;}*

9 *[C++] String\* get\_SourceTable();void set\_SourceTable(String\*);*

10 *[VB] Property SourceTable As String*

11 *[JScript] abstract function get SourceTable() : String;public abstract function set*

12 *SourceTable(String);*

13  
14 *Description*

15 *Gets or sets the case-sensitive name of the source table.*

16 *ITableMappingCollection interface (System.Data)*

17 *ToString*

18  
19  
20 *Description*

21 *Contains a collection of TableMapping objects, and is implemented by the*

22 ***System.Data.Common.DataTableMappingCollection*** , which is used in common  
23 *by .NET data providers.*

24 *The System.Data.ITableMappingCollection interface allows an inheriting*  
25 *class to implement a TableMapping collection. For more information, see .*

*Item*

*ToString*

[C#] object this[string index] {get; set;}

[C++] Object\* get\_Item(String\* index); void set\_Item(String\* index, Object\*);

[VB] Default Property Item(ByVal index As String) As Object

[JScript] abstract returnValue =

ITableMappingCollectionObject.Item(index); ITableMappingCollectionObject.Ite

m(index) = returnValue;

*Description*

*Gets or sets the instance of System.Data.ITableMapping with the specified name. The name of the System.Data.ITableMapping.*

*Add*

[C#] ITableMapping Add(string sourceTableName, string dataSetTableName);

[C++] ITableMapping\* Add(String\* sourceTableName, String\* dataSetTableName);

[VB] Function Add(ByVal sourceTableName As String, ByVal dataSetTableName As String) As ITableMapping

[JScript] function Add(sourceTableName : String, dataSetTableName : String) : ITableMapping;

*Description*

*Adds a table mapping to the collection.*

*Return Value:* A reference to the newly-mapped **System.Data.ITableMapping** object. The case-sensitive name of the source table. The name of the **System.Data.DataSetTable**.

*Contains*

*[C#] bool Contains(string sourceTableName);*

*[C++] bool Contains(String\* sourceTableName);*

*[VB] Function Contains(ByVal sourceTableName As String) As Boolean*

*[JScript] function Contains(sourceTableName : String) : Boolean;*

*Description*

*Gets a value indicating whether the collection contains a table mapping with the specified source table name.*

*Return Value:* **true** if a table mapping with the specified source table name exists, otherwise **false**. The case-sensitive name of the source table.

*GetByDataSetTable*

*[C#] ITableMapping GetByDataSetTable(string dataSetTableName);*

*[C++] ITableMapping\* GetByDataSetTable(String\* dataSetTableName);*

*[VB] Function GetByDataSetTable(ByVal dataSetTableName As String) As ITableMapping*

*[JScript] function GetByDataSetTable(dataSetTableName : String) : ITableMapping;*

## *Description*

*Gets a reference to a **System.Data.ITableMapping** table mapping.*

*Return Value: A reference to a **System.Data.ITableMapping** table mapping. The name of the **System.Data.DataSet** table within the collection.*

## *IndexOf*

*[C#] int IndexOf(string sourceTableName);*

*[C++] int IndexOf(String\* sourceTableName);*

*[VB] Function IndexOf(ByVal sourceTableName As String) As Integer*

*[JScript] function IndexOf(sourceTableName : String) : int;*

## *Description*

*Gets the location of the **System.Data.ITableMapping** object within the collection.*

*Return Value: The location of the **System.Data.ITableMapping** object within the collection. The case-sensitive name of the source table.*

## *RemoveAt*

*[C#] void RemoveAt(string sourceTableName);*

*[C++] void RemoveAt(String\* sourceTableName);*

*[VB] Sub RemoveAt(ByVal sourceTableName As String)*

*[JScript] function RemoveAt(sourceTableName : String);*

## *Description*

1 *Removes the **System.Data.ITableMapping** object with the specified name*  
2 *from the collection. The case-sensitive name of the source table.*

3 *MappingType enumeration (System.Data)*

4 *RemoveAt*

5  
6  
7 *Description*

8 *Specifies how a **System.Data.DataColumn** is mapped.*

9 *The **System.Data.MappingType** enumeration is used when getting or*  
10 *setting the **System.Data.DataColumn.ColumnMapping** property of the*  
11 ***System.Data.DataColumn** . The property determines how a column's values will*  
12 *be written when the **System.Data.DataSet.WriteXml(System.IO.Stream)** method*  
13 *is called on a **System.Data.DataSet** to write the data and schema out as an XML*  
14 *document.*

15 *RemoveAt*

16  
17 *[C#] public const MappingType Attribute;*

18 *[C++] public: const MappingType Attribute;*

19 *[VB] Public Const Attribute As MappingType*

20 *[JScript] public var Attribute : MappingType;*

21  
22 *Description*

23 *The column is mapped to an XML attribute.*

24 *RemoveAt*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const MappingType Element;
[C++] public: const MappingType Element;
[VB] Public Const Element As MappingType
[JScript] public var Element : MappingType;
```

*Description*

*The column is mapped to an XML element.*

*RemoveAt*

```
[C#] public const MappingType Hidden;
[C++] public: const MappingType Hidden;
[VB] Public Const Hidden As MappingType
[JScript] public var Hidden : MappingType;
```

*Description*

*The column is mapped to an internal structure.*

*RemoveAt*

```
[C#] public const MappingType SimpleContent;
[C++] public: const MappingType SimpleContent;
[VB] Public Const SimpleContent As MappingType
[JScript] public var SimpleContent : MappingType;
```

*Description*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

The column is mapped to an *System.Xml.XmlText* node.

*MergeFailedEventArgs* class (*System.Data*)

*ToString*

*Description*

Occurs when a target and source **DataRow** have the same primary key value, and the *System.Data.DataSet.EnforceConstraints* property is set to true.

*MergeFailedEventArgs*

*Example Syntax:*

*ToString*

[C#] public MergeFailedEventArgs(DataTable table, string conflict);

[C++] public: MergeFailedEventArgs(DataTable\* table, String\* conflict);

[VB] Public Sub New(ByVal table As DataTable, ByVal conflict As String)

[JScript] public function MergeFailedEventArgs(table : DataTable, conflict : String);

*Description*

Initializes a new instance of a *System.Data.MergeFailedEventArgs* class with the *System.Data.DataTable* name and a description of the merge conflict.

The *System.Data.DataTable* name. A description of the merge conflict.

*Conflict*

*ToString*

1  
2 *[C#] public string Conflict {get;}*  
3 *[C++] public: \_\_property String\* get\_Conflict();*  
4 *[VB] Public ReadOnly Property Conflict As String*  
5 *[JScript] public function get Conflict() : String;*  
6

7 *Description*

8 *Returns a description of the merge conflict.*

9 *Table*

10 *ToString*  
11

12 *[C#] public DataTable Table {get;}*  
13 *[C++] public: \_\_property DataTable\* get\_Table();*  
14 *[VB] Public ReadOnly Property Table As DataTable*  
15 *[JScript] public function get Table() : DataTable;*  
16

17 *Description*

18 *Returns the name of the **System.Data.DataTable** .*

19 *MergeFailedEventHandler delegate (System.Data)*

20 *ToString*  
21

22  
23 *Description*

24 *Represents the method that will handle the*

25 ***System.Data.DataSet.MergeFailed** event.*

When you create a **System.Data.MergeFailedEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

**MissingMappingAction** enumeration (System.Data)

**ToString**

#### Description

Determines the action that occurs when a mapping is missing from a source table or a source column.

The **System.Data.MissingMappingAction** values are used as arguments in the **System.Data.Common.DataColumnMappingCollection.GetColumnMappingBySchemaAction(System.Data.Common.DataColumnMappingCollection, System.String, System.Data.MissingMappingAction)** method, and the **System.Data.Common.DataTableMappingCollection.GetTableMappingBySchemaAction(System.Data.Common.DataTableMappingCollection, System.String, System.String, System.Data.MissingMappingAction)** method.

**ToString**

[C#] **public const MissingMappingAction Error;**

[C++] **public: const MissingMappingAction Error;**

[VB] **Public Const Error As MissingMappingAction**

1 *[JScript] public var Error : MissingMappingAction;*

3 *Description*

4 *A System.SystemException is generated.*

5 *ToString*

7 *[C#] public const MissingMappingAction Ignore;*

8 *[C++] public: const MissingMappingAction Ignore;*

9 *[VB] Public Const Ignore As MissingMappingAction*

10 *[JScript] public var Ignore : MissingMappingAction;*

12 *Description*

13 *The column or table not having a mapping is ignored. Returns null .*

14 *ToString*

16 *[C#] public const MissingMappingAction Passthrough;*

17 *[C++] public: const MissingMappingAction Passthrough;*

18 *[VB] Public Const Passthrough As MissingMappingAction*

19 *[JScript] public var Passthrough : MissingMappingAction;*

21 *Description*

22 *The source column or source table created and added to the*

23 *System.Data.DataSet using its original name.*

24 *MissingPrimaryKeyException class (System.Data)*

25 *ToString*

1  
2  
3 *Description*

4       *Represents the exception that is thrown when attempting to access a row in*  
5 *a table that has no primary key.*

6       *The **System.Data.MissingPrimaryKeyException** is thrown when invoking*  
7 *the following methods to access a row in a table that has no primary key.*

8       *MissingPrimaryKeyException*

9       *Example Syntax:*

10       *ToString*

11  
12 *[C#] public MissingPrimaryKeyException();*

13 *[C++] public: MissingPrimaryKeyException();*

14 *[VB] Public Sub New()*

15 *[JScript] public function MissingPrimaryKeyException();*

16  
17 *Description*

18       *Initializes a new instance of the*  
19 ***System.Data.MissingPrimaryKeyException** class.*

20       *MissingPrimaryKeyException*

21       *Example Syntax:*

22       *ToString*

23  
24 *[C#] public MissingPrimaryKeyException(string s);*

25 *[C++] public: MissingPrimaryKeyException(String\* s);*

1 *[VB] Public Sub New(ByVal s As String)*

2 *[JScript] public function MissingPrimaryKeyException(s : String);*

3  
4 *Description*

5 *Initializes a new instance of the*  
6 ***System.Data.MissingPrimaryKeyException** class with the specified string. The*  
7 *string to display when the exception is thrown.*

8 *MissingPrimaryKeyException*

9 *Example Syntax:*

10 *ToString*

11  
12 *[C#] public MissingPrimaryKeyException(SerializationInfo info,*  
13 *StreamingContext context);*

14 *[C++] public: MissingPrimaryKeyException(SerializationInfo\* info,*  
15 *StreamingContext context);*

16 *[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As*  
17 *StreamingContext)*

18 *[JScript] public function MissingPrimaryKeyException(info : SerializationInfo,*  
19 *context : StreamingContext); Initializes a new instance of the*  
20 ***System.Data.MissingPrimaryKeyException** class.*

21  
22 *Description*

23 *Initializes a new instance of the*  
24 ***System.Data.MissingPrimaryKeyException** class with serialization information.*

1 *The data necessary to serialize or deserialize an object. A description of the*  
2 *source and destination of the specified serialized stream.*

3 *HelpLink*

4 *HResult*

5 *InnerException*

6 *Message*

7 *Source*

8 *StackTrace*

9 *TargetSite*

10 *MissingSchemaAction* enumeration (*System.Data*)

11 *ToString*

12  
13  
14 *Description*

15 *Specifies the action to take when adding data to the **System.Data.DataSet***  
16 *and the required **System.Data.DataTable** or **System.Data.DataColumn** is missing.*

17 *The **System.Data.MissingSchemaAction** values are used whenever an*  
18 *action is taken that could change the schema of the **System.Data.DataSet** .*

19 *ToString*

20  
21 *[C#] public const MissingSchemaAction Add;*

22 *[C++] public: const MissingSchemaAction Add;*

23 *[VB] Public Const Add As MissingSchemaAction*

24 *[JScript] public var Add : MissingSchemaAction;*

## Description

*Adds the necessary columns to complete the schema.*

## *ToString*

*[C#] public const MissingSchemaAction AddWithKey;*

*[C++] public: const MissingSchemaAction AddWithKey;*

*[VB] Public Const AddWithKey As MissingSchemaAction*

*[JScript] public var AddWithKey : MissingSchemaAction;*

## Description

*Adds the necessary columns and primary key information to complete the schema. For more information about how primary key information is added to a*

***System.Data.DataTable** , see*

***System.Data.IDataAdapter.FillSchema(System.Data.DataSet, System.Data.SchemaType)** .To function properly with the OLE DB .NET Data Provider,*

***AddWithKey** requires that the native OLE DB provider obtains necessary primary key information by setting the DBPROP\_UNIQUEROWS property, and then*

*determines which columns are primary key columns by examining*

*DBCOLUMN\_KEYCOLUMN in the IColumnsRowset. As an alternative, the user may explicitly set the primary key constraints on each **System.Data.DataTable** .*

*This ensures that incoming records that match existing records are updated instead of appended. When using **AddWithKey** , the SQL Server .NET Data*

*Provider appends a FOR BROWSE clause to the statement being executed. The*

*user should be aware of potential side effects, such as interference with the use of*

1 *SET FMTONLY ON* statements. See *SQL Server Books Online* for more  
2 information.

3 *ToString*

4  
5 *[C#] public const MissingSchemaAction Error;*

6 *[C++] public: const MissingSchemaAction Error;*

7 *[VB] Public Const Error As MissingSchemaAction*

8 *[JScript] public var Error : MissingSchemaAction;*

9  
10 *Description*

11 *A System.SystemException is generated.*

12 *ToString*

13  
14 *[C#] public const MissingSchemaAction Ignore;*

15 *[C++] public: const MissingSchemaAction Ignore;*

16 *[VB] Public Const Ignore As MissingSchemaAction*

17 *[JScript] public var Ignore : MissingSchemaAction;*

18  
19 *Description*

20 *Ignores the extra columns.*

21 *NotNullAllowedException class (System.Data)*

22 *ToString*

23  
24  
25 *Description*

Represents the exception that is thrown when attempting to insert a null value into a column where **System.Data.DataColumn.AllowDBNull** is set to **false**

The **System.Data.NoNullAllowedException** is thrown when invoking the following methods or properties when attempting to insert a null value into a column where **System.Data.DataColumn.AllowDBNull** is set to **false**.

**NoNullAllowedException**

*Example Syntax:*

*ToString*

[C#] **public NoNullAllowedException();**

[C++] **public: NoNullAllowedException();**

[VB] **Public Sub New()**

[JScript] **public function NoNullAllowedException();**

*Description*

Initializes a new instance of the **System.Data.NoNullAllowedException** class.

**NoNullAllowedException**

*Example Syntax:*

*ToString*

[C#] **public NoNullAllowedException(string s);**

[C++] **public: NoNullAllowedException(String\* s);**

[VB] **Public Sub New(ByVal s As String)**

1 *[JScript] public function NoNullAllowedException(s : String);*

2

3 *Description*

4 *Initializes a new instance of the **System.Data.NoNullAllowedException***  
 5 *class with the specified string. The string to display when the exception is thrown.*

6 *NoNullAllowedException*

7 *Example Syntax:*

8 *ToString*

9

10 *[C#] public NoNullAllowedException(SerializationInfo info, StreamingContext*  
 11 *context);*

12 *[C++] public: NoNullAllowedException(SerializationInfo\* info,*  
 13 *StreamingContext context);*

14 *[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As*  
 15 *StreamingContext)*

16 *[JScript] public function NoNullAllowedException(info : SerializationInfo, context*  
 17 *: StreamingContext); Initializes a new instance of the*  
 18 ***System.Data.NoNullAllowedException** class.*

19

20 *Description*

21 *Initializes a new instance of the **System.Data.NoNullAllowedException***  
 22 *class with serialization information. The data necessary to serialize or deserialize*  
 23 *an object. Description of the source and destination of the specified serialized*  
 24 *stream.*

25 *HelpLink*

*HResult*

*InnerException*

*Message*

*Source*

*StackTrace*

*TargetSite*

*ParameterDirection* enumeration (*System.Data*)

*ToString*

*Description*

*Specifies the type of a parameter within a query relative to the*

***System.Data.DataSet*** .

*The **System.Data.ParameterDirection** values are used by the parameter*

*direction properties of **System.Data.OleDb.OleDbParameter** and*

***System.Data.SqlClient.SqlParameter*** .

*ToString*

*[C#] public const ParameterDirection Input;*

*[C++] public: const ParameterDirection Input;*

*[VB] Public Const Input As ParameterDirection*

*[JScript] public var Input : ParameterDirection;*

*Description*

*The parameter is an input parameter.*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*ToString*

*[C#] public const ParameterDirection InputOutput;*  
*[C++] public: const ParameterDirection InputOutput;*  
*[VB] Public Const InputOutput As ParameterDirection*  
*[JScript] public var InputOutput : ParameterDirection;*

*Description*

*The parameter is capable of both input and output.*

*ToString*

*[C#] public const ParameterDirection Output;*  
*[C++] public: const ParameterDirection Output;*  
*[VB] Public Const Output As ParameterDirection*  
*[JScript] public var Output : ParameterDirection;*

*Description*

*The parameter is an output parameter.*

*ToString*

*[C#] public const ParameterDirection ReturnValue;*  
*[C++] public: const ParameterDirection ReturnValue;*  
*[VB] Public Const ReturnValue As ParameterDirection*  
*[JScript] public var ReturnValue : ParameterDirection;*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

*The parameter represents a return value from an operation such as a stored procedure, built-in function, or user-defined function.*

*PropertyAttributes enumeration (System.Data)*  
*ToString*

*Description*

*Specifies the attributes of a property.*  
*ToString*

*[C#] public const PropertyAttributes NotSupported;*  
*[C++] public: const PropertyAttributes NotSupported;*  
*[VB] Public Const NotSupported As PropertyAttributes*  
*[JScript] public var NotSupported : PropertyAttributes;*

*Description*

*The property is not supported by the provider.*  
*ToString*

*[C#] public const PropertyAttributes Optional;*  
*[C++] public: const PropertyAttributes Optional;*  
*[VB] Public Const Optional As PropertyAttributes*  
*[JScript] public var Optional : PropertyAttributes;*

1  
2 *Description*

3       *The user does not need to specify a value for this property before the data*  
4 *source is initialized.*

5       *ToString*

6  
7 *[C#] public const PropertyAttributes Read;*

8 *[C++] public: const PropertyAttributes Read;*

9 *[VB] Public Const Read As PropertyAttributes*

10 *[JScript] public var Read : PropertyAttributes;*

11  
12 *Description*

13       *The user can read the property.*

14       *ToString*

15  
16 *[C#] public const PropertyAttributes Required;*

17 *[C++] public: const PropertyAttributes Required;*

18 *[VB] Public Const Required As PropertyAttributes*

19 *[JScript] public var Required : PropertyAttributes;*

20  
21 *Description*

22       *The user must specify a value for this property before the data source is*  
23 *initialized.*

24       *ToString*

```

1
2 [C#] public const PropertyAttributes Write;
3 [C++] public: const PropertyAttributes Write;
4 [VB] Public Const Write As PropertyAttributes
5 [JScript] public var Write : PropertyAttributes;
6

```

#### Description

The user can write to the property.

PropertyCollection class (System.Data)

ToString

#### Description

Represents a collection of properties that can be added to

**System.Data.DataColumn** , **System.Data.DataSet** , or **System.Data.DataTable** .

The **System.Data.PropertyCollection** can be accessed through the

**ExtendedProperties** property of the **System.Data.DataColumn** ,

**System.Data.DataSet** , or **System.Data.DataTable** class.

PropertyCollection

Example Syntax:

ToString

```

23 [C#] public PropertyCollection();
24 [C++] public: PropertyCollection();
25

```

1	<i>[VB] Public Sub New()</i>
2	<i>[JScript] public function PropertyCollection();</i>
3	<i>comparer</i>
4	<i>Count</i>
5	<i>hcp</i>
6	<i>IsFixedSize</i>
7	<i>IsReadOnly</i>
8	<i>IsSynchronized</i>
9	<i>Item</i>
10	<i>Keys</i>
11	<i>SyncRoot</i>
12	<i>Values</i>
13	<i>ReadOnlyException class (System.Data)</i>
14	<i>ToString</i>
15	
16	
17	<i>Description</i>
18	<i>Represents the exception that is thrown when attempting to change the</i>
19	<i>value of a read-only column.</i>
20	<i>The <b>System.Data.RowNotInTableException</b> is thrown when invoking the</i>
21	<i>following methods or properties when attempting to change the value of a read-</i>
22	<i>only column.</i>
23	<i>ReadOnlyException</i>
24	<i>Example Syntax:</i>
25	<i>ToString</i>

[C#] *public ReadOnlyException();*

[C++] *public: ReadOnlyException();*

[VB] *Public Sub New()*

[JScript] *public function ReadOnlyException();*

### *Description*

*Initializes a new instance of the **System.Data.ReadOnlyException** class.*

*ReadOnlyException*

*Example Syntax:*

*ToString*

[C#] *public ReadOnlyException(string s);*

[C++] *public: ReadOnlyException(String\* s);*

[VB] *Public Sub New(ByVal s As String)*

[JScript] *public function ReadOnlyException(s : String);*

### *Description*

*Initializes a new instance of the **System.Data.ReadOnlyException** class with the specified string. The string to display when the exception is thrown.*

*ReadOnlyException*

*Example Syntax:*

*ToString*

[C#] *public ReadOnlyException(SerializationInfo info, StreamingContext*

```

1 context);
2 [C++] public: ReadOnlyException(SerializationInfo* info, StreamingContext
3 context);
4 [VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As
5 StreamingContext)
6 [JScript] public function ReadOnlyException(info : SerializationInfo, context :
7 StreamingContext); Initializes a new instance of the
8 System.Data.ReadOnlyException class.

```

#### Description

Initializes a new instance of the **System.Data.ReadOnlyException** class with serialization information. The data necessary to serialize or deserialize an object. Description of the source and destination of the specified serialized stream.

*HelpLink*

*HResult*

*InnerException*

*Message*

*Source*

*StackTrace*

*TargetSite*

*RowNotInTableException* class (System.Data)

*ToString*

#### Description

1       Represents the exception that is thrown when trying to perform an  
2 operation on a **System.Data.DataRow** that is not in a **System.Data.DataTable** .

3       The **System.Data.RowNotFoundException** is thrown when invoking the  
4 following methods on a row that has been deleted with either the  
5 **System.Data.DataRow.Delete** or the  
6 **System.Data.DataRowCollection.Remove(System.Data.DataRow)** method.

7       RowNotFoundException

8       Example Syntax:

9       ToString

10  
11 [C#] public RowNotFoundException();

12 [C++] public: RowNotFoundException();

13 [VB] Public Sub New()

14 [JScript] public function RowNotFoundException(); Initializes a new instance of  
15 the **System.Data.RowNotFoundException** class with no arguments.

16  
17 Description

18       Initializes a new instance of the **System.Data.RowNotFoundException**  
19 class.

20       RowNotFoundException

21       Example Syntax:

22       ToString

23  
24 [C#] public RowNotFoundException(string s);

25 [C++] public: RowNotFoundException(String\* s);

1 *[VB] Public Sub New(ByVal s As String)*

2 *[JScript] public function RowNotInTableException(s : String);*

3  
4 *Description*

5 *Initializes a new instance of the **System.Data.RowNotInTableException***  
6 *class with the specified string. The string to display when the exception is thrown.*

7 *RowNotInTableException*

8 *Example Syntax:*

9 *ToString*

10  
11 *[C#] public RowNotInTableException(SerializationInfo info, StreamingContext*  
12 *context);*

13 *[C++] public: RowNotInTableException(SerializationInfo\* info,*  
14 *StreamingContext context);*

15 *[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As*  
16 *StreamingContext)*

17 *[JScript] public function RowNotInTableException(info : SerializationInfo,*  
18 *context : StreamingContext);* *Initializes a new instance of the*

19 ***System.Data.RowNotInTableException** class.*

20  
21 *Description*

22 *Initializes a new instance of the **System.Data.RowNotInTableException***  
23 *class with serialization information. The data necessary to serialize or deserialize*  
24 *an object. Description of the source and destination of the specified serialized*  
25 *stream.*

1	<i>HelpLink</i>
2	<i>HResult</i>
3	<i>InnerException</i>
4	<i>Message</i>
5	<i>Source</i>
6	<i>StackTrace</i>
7	<i>TargetSite</i>
8	<i>Rule enumeration (System.Data)</i>
9	<i>ToString</i>
10	
11	
12	<i>Description</i>
13	<i>Indicates the action that occurs when a</i>
14	<b><i>System.Data.ForeignKeyConstraint</i></b> <i>is enforced.</i>
15	<i>The System.Data.Rule values are set to the</i>
16	<b><i>System.Data.ForeignKeyConstraint.UpdateRule</i></b> <i>and the</i>
17	<b><i>System.Data.ForeignKeyConstraint.DeleteRule</i></b> <i>properties of a</i>
18	<b><i>System.Data.ForeignKeyConstraint</i></b> <i>object found in a System.Data.DataTable</i>
19	<i>object's System.Data.ConstraintCollection .</i>
20	<i>ToString</i>
21	
22	<i>[C#] public const Rule Cascade;</i>
23	<i>[C++] public: const Rule Cascade;</i>
24	<i>[VB] Public Const Cascade As Rule</i>
25	<i>[JScript] public var Cascade : Rule;</i>

*Description**Delete or update related rows. This is the default.**ToString**[C#] public const Rule None;**[C++] public: const Rule None;**[VB] Public Const None As Rule**[JScript] public var None : Rule;**Description**No action taken on related rows.**ToString**[C#] public const Rule SetDefault;**[C++] public: const Rule SetDefault;**[VB] Public Const SetDefault As Rule**[JScript] public var SetDefault : Rule;**Description**Set values in related rows to the value contained in the**System.Data.DataColumn.DefaultValue property.**ToString**[C#] public const Rule SetNull;*

1 *[C++] public: const Rule SetNull;*  
 2 *[VB] Public Const SetNull As Rule*  
 3 *[JScript] public var SetNull : Rule;*  
 4

5 *Description*

6 *Set values in related rows to DBNull .*  
 7 *SchemaType enumeration (System.Data)*  
 8 *ToString*  
 9  
 10

11 *Description*

12 *Specifies how to handle existing schema mappings when performing a*  
 13 *System.Data.Common.DataAdapter.FillSchema(System.Data.DataSet, System.D*  
 14 *ata.SchemaType) operation.*

15 *The System.Data.SchemaType usually should be set to Mapped , because*  
 16 *any established table and column mappings are used.*

17 *ToString*  
 18

19 *[C#] public const SchemaType Mapped;*  
 20 *[C++] public: const SchemaType Mapped;*  
 21 *[VB] Public Const Mapped As SchemaType*  
 22 *[JScript] public var Mapped : SchemaType;*  
 23

24 *Description*  
 25

1 *Apply any existing table mappings to the incoming schema. Configure the*  
2 ***System.Data.DataSet** with the transformed schema.*

3 *ToString*

4  
5 *[C#] public const SchemaType Source;*

6 *[C++] public: const SchemaType Source;*

7 *[VB] Public Const Source As SchemaType*

8 *[JScript] public var Source : SchemaType;*

9  
10 *Description*

11 *Ignore any table mappings on the DataAdapter. Configure the*  
12 ***System.Data.DataSet** using the incoming schema without applying any*  
13 *transformations.*

14 *SqlDbType enumeration (System.Data)*

15 *ToString*

16  
17  
18 *Description*

19 *Specifies SQL Server data types.*

20 *ToString*

21  
22 *[C#] public const SqlDbType BigInt;*

23 *[C++] public: const SqlDbType BigInt;*

24 *[VB] Public Const BigInt As SqlDbType*

25 *[JScript] public var BigInt : SqlDbType;*

1  
2 *Description*

3 ***System.Int64** A 64-bit signed integer.*

4 *ToString*

5  
6 *[C#] public const SqlDbType Binary;*

7 *[C++] public: const SqlDbType Binary;*

8 *[VB] Public Const Binary As SqlDbType*

9 *[JScript] public var Binary : SqlDbType;*

10  
11 *Description*

12 ***System.Array** of type **System.Byte** A fixed-length stream of binary data*  
13 *ranging between 1 and 8,000 bytes.*

14 *ToString*

15  
16 *[C#] public const SqlDbType Bit;*

17 *[C++] public: const SqlDbType Bit;*

18 *[VB] Public Const Bit As SqlDbType*

19 *[JScript] public var Bit : SqlDbType;*

20  
21 *Description*

22 ***System.Boolean** An unsigned numeric value that can be 0, 1, or **null** .*

23 *ToString*

24  
25 *[C#] public const SqlDbType Char;*

1 *[C++] public: const SqlDbType Char;*  
 2 *[VB] Public Const Char As SqlDbType*  
 3 *[JScript] public var Char : SqlDbType;*

4  
 5 *Description*

6 *System.String A fixed-length stream of non-Unicode characters ranging*  
 7 *between 1 and 8,000 characters.*

8 *ToString*

9  
 10 *[C#] public const SqlDbType DateTime;*  
 11 *[C++] public: const SqlDbType DateTime;*  
 12 *[VB] Public Const DateTime As SqlDbType*  
 13 *[JScript] public var DateTime : SqlDbType;*

14  
 15 *Description*

16 *System.DateTime Date and time data ranging in value from January 1,*  
 17 *1753 to December 31, 9999 to an accuracy of 3.33 milliseconds.*

18 *ToString*

19  
 20 *[C#] public const SqlDbType Decimal;*  
 21 *[C++] public: const SqlDbType Decimal;*  
 22 *[VB] Public Const Decimal As SqlDbType*  
 23 *[JScript] public var Decimal : SqlDbType;*

24  
 25 *Description*

**System.Decimal** A fixed precision and scale numeric value between  $-10^{-1}$  and  $10^{-1}$ .

*ToString*

[C#] public const SqlDbType Float;

[C++] public: const SqlDbType Float;

[VB] Public Const Float As SqlDbType

[JScript] public var Float : SqlDbType;

*Description*

**System.Double** A floating point number within the range of  $-1.79E + 308$  through  $1.79E + 308$ .

*ToString*

[C#] public const SqlDbType Image;

[C++] public: const SqlDbType Image;

[VB] Public Const Image As SqlDbType

[JScript] public var Image : SqlDbType;

*Description*

**System.Array** of type **System.Byte** A variable-length stream of binary data ranging from 0 to  $2^{31} - 1$  (or 2,147,483,647) bytes.

*ToString*

[C#] public const SqlDbType Int;

1 *[C++] public: const SqlDbType Int;*  
 2 *[VB] Public Const Int As SqlDbType*  
 3 *[JScript] public var Int : SqlDbType;*

4  
 5 *Description*

6 ***System.Int32** A 32-bit signed integer.*

7 *ToString*

8  
 9 *[C#] public const SqlDbType Money;*  
 10 *[C++] public: const SqlDbType Money;*  
 11 *[VB] Public Const Money As SqlDbType*  
 12 *[JScript] public var Money : SqlDbType;*

13  
 14 *Description*

15 ***System.Decimal** A currency value ranging from -2 (or -*  
 16 *922,337,203,685,477.5808) to 2 -1 (or +922,337,203,685,477.5807) with an*  
 17 *accuracy to a ten-thousandth of a currency unit.*

18 *ToString*

19  
 20 *[C#] public const SqlDbType NChar;*  
 21 *[C++] public: const SqlDbType NChar;*  
 22 *[VB] Public Const NChar As SqlDbType*  
 23 *[JScript] public var NChar : SqlDbType;*

24  
 25 *Description*

**System.String** *A fixed-length stream of Unicode characters ranging between 1 and 4,000 characters.*

*ToString*

[C#] *public const SqlDbType NText;*

[C++] *public: const SqlDbType NText;*

[VB] *Public Const NText As SqlDbType*

[JScript] *public var NText : SqlDbType;*

*Description*

**System.String** *A variable-length stream of Unicode data with a maximum length of 2 - 1 (or 1,073,741,823) characters.*

*ToString*

[C#] *public const SqlDbType NVarChar;*

[C++] *public: const SqlDbType NVarChar;*

[VB] *Public Const NVarChar As SqlDbType*

[JScript] *public var NVarChar : SqlDbType;*

*Description*

**System.String** *A variable-length stream of Unicode characters ranging between 1 and 4,000 characters.*

*ToString*

[C#] *public const SqlDbType Real;*

1 *[C++] public: const SqlDbType Real;*

2 *[VB] Public Const Real As SqlDbType*

3 *[JScript] public var Real : SqlDbType;*

4

5 *Description*

6 *System.Single A floating point number within the range of -3.40E +38*  
 7 *through 3.40E +38.*

8 *ToString*

9

10 *[C#] public const SqlDbType SmallDateTime;*

11 *[C++] public: const SqlDbType SmallDateTime;*

12 *[VB] Public Const SmallDateTime As SqlDbType*

13 *[JScript] public var SmallDateTime : SqlDbType;*

14

15 *Description*

16 *System.DateTime Date and time data ranging in value from January 1,*  
 17 *1900 to June 6, 2079 to an accuracy of one minute.*

18 *ToString*

19

20 *[C#] public const SqlDbType SmallInt;*

21 *[C++] public: const SqlDbType SmallInt;*

22 *[VB] Public Const SmallInt As SqlDbType*

23 *[JScript] public var SmallInt : SqlDbType;*

24

25 *Description*

***System.Int16*** A 16-bit signed integer.

*ToString*

[C#] public const SqlDbType SmallMoney;

[C++] public: const SqlDbType SmallMoney;

[VB] Public Const SmallMoney As SqlDbType

[JScript] public var SmallMoney : SqlDbType;

*Description*

***System.Decimal*** A currency value ranging from -214,748.3648 to +214,748.3647 with an accuracy to a ten-thousandth of a currency unit.

*ToString*

[C#] public const SqlDbType Text;

[C++] public: const SqlDbType Text;

[VB] Public Const Text As SqlDbType

[JScript] public var Text : SqlDbType;

*Description*

***System.String*** A variable-length stream of non-Unicode data with a maximum length of 2<sup>31</sup> - 1 (or 2,147,483,647) characters.

*ToString*

[C#] public const SqlDbType Timestamp;

[C++] public: const SqlDbType Timestamp;

1 *[VB] Public Const Timestamp As SqlDbType*  
 2 *[JScript] public var Timestamp : SqlDbType;*

4 *Description*

5 ***System.DateTime*** Data and time data in the format *yyyymmddhhmmss*.

6 *ToString*

8 *[C#] public const SqlDbType TinyInt;*  
 9 *[C++] public: const SqlDbType TinyInt;*  
 10 *[VB] Public Const TinyInt As SqlDbType*  
 11 *[JScript] public var TinyInt : SqlDbType;*

13 *Description*

14 ***System.Byte*** An 8-bit unsigned integer.

15 *ToString*

17 *[C#] public const SqlDbType UniqueIdentifier;*  
 18 *[C++] public: const SqlDbType UniqueIdentifier;*  
 19 *[VB] Public Const UniqueIdentifier As SqlDbType*  
 20 *[JScript] public var UniqueIdentifier : SqlDbType;*

22 *Description*

23 ***System.Guid*** A globally unique identifier (or GUID).

24 *ToString*

25

```

1
2 [C#] public const SqlDbType VarBinary;
3 [C++] public: const SqlDbType VarBinary;
4 [VB] Public Const VarBinary As SqlDbType
5 [JScript] public var VarBinary : SqlDbType;

```

#### Description

**System.Array** of type **System.Byte** A variable-length stream of binary data ranging between 1 and 8,000 bytes.

#### ToString

```

12 [C#] public const SqlDbType VarChar;
13 [C++] public: const SqlDbType VarChar;
14 [VB] Public Const VarChar As SqlDbType
15 [JScript] public var VarChar : SqlDbType;

```

#### Description

**System.String** A variable-length stream of non-Unicode characters ranging between 1 and 8,000 characters.

#### ToString

```

22 [C#] public const SqlDbType Variant;
23 [C++] public: const SqlDbType Variant;
24 [VB] Public Const Variant As SqlDbType
25 [JScript] public var Variant : SqlDbType;

```

1  
2 *Description*

3       **System.Object** *A special data type that can contain numeric, string, binary,*  
4 *or date data as well as the SQL Server values Empty and Null, which is assumed if*  
5 *no other type is declared.*

6       *StateChangeEventArgs* class (*System.Data*)

7       *ToString*

8  
9  
10 *Description*

11       *Provides data for the state change event of a .NET data provider.*

12       *The data is used by the*

13       **System.Data.OleDb.OleDbConnection.StateChange** *property of the*

14       **System.Data.OleDb.OleDbConnection** *and the*

15       **System.Data.SqlClient.SqlConnection.StateChange** *property of the*

16       **System.Data.SqlClient.SqlConnection** .

17       *StateChangeEventArgs*

18       *Example Syntax:*

19       *ToString*

20  
21 *[C#] public StateChangeEventArgs(ConnectionState originalState,*

22 *ConnectionState currentState);*

23 *[C++] public: StateChangeEventArgs(ConnectionState originalState,*

24 *ConnectionState currentState);*

25 *[VB] Public Sub New(ByVal originalState As ConnectionState, ByVal currentState*

As ConnectionState)

[JScript] public function StateChangeEventArgs(originalState : ConnectionState,  
currentState : ConnectionState);

#### Description

Initializes a new instance of the **System.Data.StateChangeEventArgs** class,  
when given the original state and the current state of the object. One of the  
**System.Data.ConnectionState** values. One of the **System.Data.ConnectionState**  
values.

CurrentState

ToString

[C#] public ConnectionState CurrentState {get;}

[C++] public: \_\_property ConnectionState get\_CurrentState();

[VB] Public ReadOnly Property CurrentState As ConnectionState

[JScript] public function get CurrentState() : ConnectionState;

#### Description

Gets the new state of the connection. The connection object will be in the  
new state already when the event is fired.

OriginalState

ToString

[C#] public ConnectionState OriginalState {get;}

[C++] public: \_\_property ConnectionState get\_OriginalState();

1 *[VB] Public ReadOnly Property OriginalState As ConnectionState*

2 *[JScript] public function get OriginalState() : ConnectionState;*

3  
4 *Description*

5 *Gets the original state of the connection.*

6 *StateChangeEventHandler delegate (System.Data)*

7 *ToString*

8  
9  
10 *Description*

11 *Represents the method that will handle the*

12 *System.Data.OleDb.OleDbConnection.StateChange event. The source of the*  
13 *event. The System.Data.StateChangeEventArgs that contains the event data.*

14 *When you create a System.Data.StateChangeEventHandler delegate, you*  
15 *identify the method that will handle the event. To associate the event with your*  
16 *event handler, add an instance of the delegate to the event. The event handler is*  
17 *called whenever the event occurs, unless you remove the delegate. For more*  
18 *information about event handler delegates, see .*

19 *StatementType enumeration (System.Data)*

20 *ToString*

21  
22  
23 *Description*

24 *Specifies the type of SQL query to be used by the*

25 *System.Data.OleDb.OleDbRowUpdatedEventArgs ,*

*System.Data.OleDb.OleDbRowUpdatingEventArgs ,  
System.Data.SqlClient.SqlRowUpdatedEventArgs , or  
System.Data.SqlClient.SqlRowUpdatingEventArgs class.*

*ToString*

*[C#] public const StatementType Delete;  
[C++] public: const StatementType Delete;  
[VB] Public Const Delete As StatementType  
[JScript] public var Delete : StatementType;*

*Description*

*A SQL query that is a DELETE statement.*

*ToString*

*[C#] public const StatementType Insert;  
[C++] public: const StatementType Insert;  
[VB] Public Const Insert As StatementType  
[JScript] public var Insert : StatementType;*

*Description*

*A SQL query that is an INSERT statement.*

*ToString*

*[C#] public const StatementType Select;  
[C++] public: const StatementType Select;*

1 *[VB] Public Const Select As StatementType*

2 *[JScript] public var Select : StatementType;*

4 *Description*

5 *A SQL query that is a SELECT statement.*

6 *ToString*

8 *[C#] public const StatementType Update;*

9 *[C++] public: const StatementType Update;*

10 *[VB] Public Const Update As StatementType*

11 *[JScript] public var Update : StatementType;*

13 *Description*

14 *A SQL query that is an UPDATE statement.*

15 *StrongTypingException class (System.Data)*

16 *ToString*

19 *Description*

20 *The exception that is thrown by a strongly-typed System.Data.DataSet*  
 21 *when the user accesses DBNull value.*

22 *The System.Data.StrongTypingException class is not intended for use as a*  
 23 *stand alone component, but as a class from which other classes derive standard*  
 24 *functionality.*

25 *StrongTypingException*

*Example Syntax:*

*ToString*

*[C#] public StrongTypingException();*

*[C++] public: StrongTypingException();*

*[VB] Public Sub New()*

*[JScript] public function StrongTypingException(); Initializes a new instance of the **System.Data.StrongTypingException** class.*

#### *Description*

*Initializes a new instance of the **System.Data.StrongTypingException** class.*

*The **System.Data.StrongTypingException** class is not intended for use as a stand alone component, but as a class from which other classes derive standard functionality.*

*StrongTypingException*

*Example Syntax:*

*ToString*

*[C#] public StrongTypingException(SerializationInfo info, StreamingContext context);*

*[C++] public: StrongTypingException(SerializationInfo\* info, StreamingContext context);*

*[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)*

1 *[JScript] public function StrongTypingException(info : SerializationInfo, context :  
2 StreamingContext);*

3 *StrongTypingException*

4 *Example Syntax:*

5 *ToString*

6  
7 *[C#] public StrongTypingException(string s, Exception innerException);*

8 *[C++] public: StrongTypingException(String\* s, Exception\* innerException);*

9 *[VB] Public Sub New(ByVal s As String, ByVal innerException As Exception)*

10 *[JScript] public function StrongTypingException(s : String, innerException :  
11 Exception);*

12  
13 *Description*

14 *Initializes a new instance of the **System.Data.StrongTypingException** class*  
15 *with the specified string and inner exception.*

16 *The **System.Data.StrongTypingException** class is not intended for use as a*  
17 *stand alone component, but as a class from which other classes derive standard*  
18 *functionality. The string to display when the exception is thrown. A reference to an*  
19 *inner exception.*

20 *HelpLink*

21 *HResult*

22 *InnerException*

23 *Message*

24 *Source*

25 *StackTrace*

*TargetSite*

*SyntaxErrorException class (System.Data)*

*ToString*

*Description*

*Represents the exception that is thrown when the **System.Data.DataColumn.Expression** property of a **System.Data.DataColumn** contains a syntax error.*

*SyntaxErrorException*

*Example Syntax:*

*ToString*

*[C#] public SyntaxErrorException();*

*[C++] public: SyntaxErrorException();*

*[VB] Public Sub New()*

*[JScript] public function SyntaxErrorException(); Initializes a new instance of the **System.Data.SyntaxErrorException** class.*

*Description*

*Initializes a new instance of the **System.Data.SyntaxErrorException** class.*

*SyntaxErrorException*

*Example Syntax:*

*ToString*

```

1
2 [C#] public SyntaxErrorException(string s);
3 [C++] public: SyntaxErrorException(String* s);
4 [VB] Public Sub New(ByVal s As String)
5 [JScript] public function SyntaxErrorException(s : String);

```

#### Description

Initializes a new instance of the **System.Data.SyntaxErrorException** class with the specified string. The string to display when the exception is thrown.

**SyntaxErrorException**

*Example Syntax:*

*ToString*

```

14 [C#] public SyntaxErrorException(SerializationInfo info, StreamingContext
15 context);
16 [C++] public: SyntaxErrorException(SerializationInfo* info, StreamingContext
17 context);
18 [VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As
19 StreamingContext)
20 [JScript] public function SyntaxErrorException(info : SerializationInfo, context :
21 StreamingContext);

```

#### Description

Initializes a new instance of the **System.Data.SyntaxErrorException** class with the **System.Runtime.Serialization.SerializationInfo** and the

***System.Runtime.Serialization.StreamingContext*** . The data needed to serialize or deserialize an object. The source and destination of a given serialized stream.

*HelpLink*

*HResult*

*InnerException*

*Message*

*Source*

*StackTrace*

*TargetSite*

*TypedDataSetGenerator* class (System.Data)

*ToString*

#### ***Description***

*Used to create a strongly-typed System.Data.DataSet .*

*The System.Data.TypedDataSetGenerator class is not intended for use as a stand alone component, but as a class from which other classes derive standard functionality.*

*TypedDataSetGenerator*

*Example Syntax:*

*ToString*

***[C#] public TypedDataSetGenerator();***

***[C++] public: TypedDataSetGenerator();***

1 *[VB] Public Sub New()*

2 *[JScript] public function TypedDataSetGenerator();*

3 *Generate*

4  
5 *[C#] public static void Generate(DataSet dataSet, CodeNamespace*  
6 *codeNamespace, ICodeGenerator codeGen);*

7 *[C++] public: static void Generate(DataSet\* dataSet, CodeNamespace\**  
8 *codeNamespace, ICodeGenerator\* codeGen);*

9 *[VB] Public Shared Sub Generate(ByVal dataSet As DataSet, ByVal*  
10 *codeNamespace As CodeNamespace, ByVal codeGen As ICodeGenerator)*

11 *[JScript] public static function Generate(dataSet : DataSet, codeNamespace :*  
12 *CodeNamespace, codeGen : ICodeGenerator); Generates a strongly-typed*  
13 ***System.Data.DataSet** .*

14  
15 *Description*

16 *Generates a strongly-typed **System.Data.DataSet** .*

17 *The **System.Data.TypedDataSetGenerator** class is not intended for use as a*  
18 *stand alone component, but as a class from which other classes derive standard*  
19 *functionality. The source **System.Data.DataSet** that specifies the metadata for the*  
20 *typed **System.Data.DataSet** . The **CodeNamespace** that provides the target*  
21 ***Namespace** for the typed **System.Data.DataSet** . The **CodeGenerator** used to*  
22 *create the typed **System.Data.DataSet** .*

23 *GenerateIdName*

24  
25 *[C#] public static string GenerateIdName(string name, ICodeGenerator*

1 *codeGen);*

2 *[C++] public: static String\* GenerateIdName(String\* name, ICodeGenerator\**  
3 *codeGen);*

4 *[VB] Public Shared Function GenerateIdName(ByVal name As String, ByVal*  
5 *codeGen As ICodeGenerator) As String*

6 *[JScript] public static function GenerateIdName(name : String, codeGen :*  
7 *ICodeGenerator) : String;*

### 9 *Description*

10 *Transforms a string in a valid typed **System.Data.DataSet** name.*

11 *Return Value: A string that is the converted name.*

12 *The **System.Data.TypedDataSetGenerator** class is not intended for use as a*  
13 *stand alone component, but as a class from which other classes derive standard*  
14 *functionality. The source name to transform into a valid typed*  
15 ***System.Data.DataSet** name. The CodeGenerator used to perform the conversion.*

16 *TypedDataSetGeneratorException class (System.Data)*

17 *ToString*

### 20 *Description*

21 *The exception that is thrown when a name conflict occurs while generating*  
22 *a strongly-typed **System.Data.DataSet** .*

23 *The **System.Data.TypedDataSetGeneratorException** class is not intended*  
24 *for use as a stand alone component, but as a class from which other classes derive*  
25 *standard functionality.*

*TypedDataSetGeneratorException*

*Example Syntax:*

*ToString*

*[C#] public TypedDataSetGeneratorException();*

*[C++] public: TypedDataSetGeneratorException();*

*[VB] Public Sub New()*

*[JScript] public function TypedDataSetGeneratorException(); Initializes a new instance of the **System.Data.TypedDataSetGeneratorException** class.*

*Description*

*Initializes a new instance of the **System.Data.TypedDataSetGeneratorException** class.*

*The **System.Data.TypedDataSetGeneratorException** class is not intended for use as a stand alone component, but as a class from which other classes derive standard functionality.*

*TypedDataSetGeneratorException*

*Example Syntax:*

*ToString*

*[C#] public TypedDataSetGeneratorException(ArrayList list);*

*[C++] public: TypedDataSetGeneratorException(ArrayList\* list);*

*[VB] Public Sub New(ByVal list As ArrayList)*

*[JScript] public function TypedDataSetGeneratorException(list : ArrayList);*

## *Description*

*Initializes a new instance of the*

***System.Data.TypedDataSetGeneratorException*** class.

*The System.Data.TypedDataSetGeneratorException class is not intended for use as a stand alone component, but as a class from which other classes derive standard functionality. A dynamic list of exceptions.*

*TypedDataSetGeneratorException*

*Example Syntax:*

*ToString*

*[C#] public TypedDataSetGeneratorException(SerializationInfo info, StreamingContext context);*

*[C++] public: TypedDataSetGeneratorException(SerializationInfo\* info, StreamingContext context);*

*[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)*

*[JScript] public function TypedDataSetGeneratorException(info : SerializationInfo, context : StreamingContext);*

*ErrorList*

*ToString*

*[C#] public ArrayList ErrorList {get;}*

*[C++] public: \_\_property ArrayList\* get\_ErrorList();*

*[VB] Public ReadOnly Property ErrorList As ArrayList*

1 *[JScript] public function get ErrorList() : ArrayList;*

3 *Description*

4 *Gets a dynamic list of generated errors.*

5 *The **System.Data.TypedDataSetGeneratorException** class is not intended*  
6 *for use as a stand alone component, but as a class from which other classes derive*  
7 *standard functionality.*

8 *HelpLink*

9 *HResult*

10 *InnerException*

11 *Message*

12 *Source*

13 *StackTrace*

14 *TargetSite*

15 *GetObjectData*

17 *[C#] public override void GetObjectData(SerializationInfo info,*  
18 *StreamingContext context);*

19 *[C++] public: void GetObjectData(SerializationInfo\* info, StreamingContext*  
20 *context);*

21 *[VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal*  
22 *context As StreamingContext)*

23 *[JScript] public override function GetObjectData(info : SerializationInfo, context*  
24 *: StreamingContext);*

25 *UniqueConstraint class (System.Data)*

*ToString*

*Description*

*Represents a restriction on a set of columns in which all values must be unique.*

*The **System.Data.UniqueConstraint** is enforced on a single column (or columns) to ensure that a primary key value is unique.*

*UniqueConstraint*

*Example Syntax:*

*ToString*

*[C#] public UniqueConstraint(DataColumn column);*

*[C++] public: UniqueConstraint(DataColumn\* column);*

*[VB] Public Sub New(ByVal column As DataColumn)*

*[JScript] public function UniqueConstraint(column : DataColumn);*

*Description*

*Initializes a new instance of the **System.Data.UniqueConstraint** with the specified **System.Data.DataColumn** . The **System.Data.DataColumn** to constrain.*

*UniqueConstraint*

*Example Syntax:*

*ToString*

*[C#] public UniqueConstraint(DataColumn[] columns);*

1 *[C++] public: UniqueConstraint(DataColumn\* columns[]);*

2 *[VB] Public Sub New(ByVal columns() As DataColumn)*

3 *[JScript] public function UniqueConstraint(columns : DataColumn[]);*

4  
5 *Description*

6 *Initializes a new instance of the **System.Data.UniqueConstraint** with the*  
7 *given array of **System.Data.DataColumn** objects. The array of*  
8 ***System.Data.DataColumn** objects to constrain.*

9 *UniqueConstraint*

10 *Example Syntax:*

11 *ToString*

12  
13 *[C#] public UniqueConstraint(string name, DataColumn column);*

14 *[C++] public: UniqueConstraint(String\* name, DataColumn\* column);*

15 *[VB] Public Sub New(ByVal name As String, ByVal column As DataColumn)*

16 *[JScript] public function UniqueConstraint(name : String, column :*

17 *DataColumn); Initializes a new instance of the **System.Data.UniqueConstraint** .*

18  
19 *Description*

20 *Initializes a new instance of the **System.Data.UniqueConstraint** with the*  
21 *specified name and **System.Data.DataColumn** . The name of the constraint. The*  
22 ***System.Data.DataColumn** to constrain.*

23 *UniqueConstraint*

24 *Example Syntax:*

25 *ToString*

```

1
2 [C#] public UniqueConstraint(string name, DataColumn[] columns);
3 [C++] public: UniqueConstraint(String* name, DataColumn* columns[]);
4 [VB] Public Sub New(ByVal name As String, ByVal columns() As DataColumn)
5 [JScript] public function UniqueConstraint(name : String, columns :
6 DataColumn[]);

```

#### Description

Initializes a new instance of the **System.Data.UniqueConstraint** with the specified name and array of **System.Data.DataColumn** objects. The name of the constraint. The array of **System.Data.DataColumn** objects to constrain.

*UniqueConstraint*

*Example Syntax:*

*ToString*

```

16 [C#] public UniqueConstraint(string name, string[] columnNames, bool
17 isPrimaryKey);
18 [C++] public: UniqueConstraint(String* name, String* columnNames __gc[],
19 bool isPrimaryKey);
20 [VB] Public Sub New(ByVal name As String, ByVal columnNames() As String,
21 ByVal isPrimaryKey As Boolean)
22 [JScript] public function UniqueConstraint(name : String, columnNames :
23 String[], isPrimaryKey : Boolean);

```

#### Description

1        *Initializes a new instance of the **System.Data.UniqueConstraint** with the*  
2        *specified name, an array of **System.Data.DataColumn** objects, and a value*  
3        *specifying whether the constraint is a primary key. The name of the constraint. An*  
4        *array containing names of **System.Data.DataColumn** objects to constrain.*

5        *\_DataSet*

6        *Columns*

7        *ToString*

9  
10      *Description*

11        *Gets the array of columns that this constraint affects.*

12        *ConstraintName*

13        *ExtendedProperties*

14        *IsPrimaryKey*

15        *ToString*

16  
17  
18      *Description*

19        *Gets a value indicating whether or not the constraint is on a primary key.*

20        *A table usually includes a primary key that ensures every row is unique. In*  
21        *some tables, the primary key may be made up of more than one column. For*  
22        *example, a primary key for a table containing names might be made up of both the*  
23        *first and last names as well. To create a primary key with more than one column,*  
24        *set the **Columns** property to an array of **DataColumn** objects.*

25        *Table*

## *ToString*

*[C#] public override DataTable Table {get;}*

*[C++] public: \_\_property virtual DataTable\* get\_Table();*

*[VB] Overrides Public ReadOnly Property Table As DataTable*

*[JScript] public function get Table() : DataTable;*

## *Description*

*Gets the table to which this constraint belongs.*

## *Equals*

*[C#] public override bool Equals(object key2);*

*[C++] public: bool Equals(Object\* key2);*

*[VB] Overrides Public Function Equals(ByVal key2 As Object) As Boolean*

*[JScript] public override function Equals(key2 : Object) : Boolean;*

## *Description*

*Compares this constraint to a second to determine if both are identical.*

*Return Value: **true** , if the constraints are equal; otherwise, **false** .*

*Two constraints are equal if they constrain the same columns. The object to which this **System.Data.UniqueConstraint** is compared.*

## *GetHashCode*

*[C#] public override int GetHashCode();*

*[C++] public: int GetHashCode();*

1 *[VB] Overrides Public Function GetHashCode() As Integer*  
 2 *[JScript] public override function GetHashCode() : int;*

3  
 4 *Description*  
 5 *Gets the hash code of this instance of the **System.Data.UniqueConstraint***  
 6 *object.*  
 7 *Return Value: A 32-bit signed integer hash code.*  
 8 *UpdateRowSource enumeration (System.Data)*  
 9 *ToString*

10  
 11  
 12 *Description*  
 13 *Specifies how query command results are applied to the row being updated.*  
 14 *The **System.Data.UpdateRowSource** values are used by the*  
 15 ***System.Data.IDbCommand.UpdatedRowSource** property of*  
 16 ***System.Data.IDbCommand** and any classes derived from it.*  
 17 *ToString*

18  
 19 *[C#] public const UpdateRowSource Both;*  
 20 *[C++] public: const UpdateRowSource Both;*  
 21 *[VB] Public Const Both As UpdateRowSource*  
 22 *[JScript] public var Both : UpdateRowSource;*

23  
 24 *Description*  
 25

Both the output parameters and the first returned row are mapped to the changed row in the **System.Data.DataSet** .

*ToString*

```
[C#] public const UpdateRowSource FirstReturnedRecord;  
[C++] public: const UpdateRowSource FirstReturnedRecord;  
[VB] Public Const FirstReturnedRecord As UpdateRowSource  
[JScript] public var FirstReturnedRecord : UpdateRowSource;
```

*Description*

The data in the first returned row is mapped to the changed row in the **System.Data.DataSet** .

*ToString*

```
[C#] public const UpdateRowSource None;  
[C++] public: const UpdateRowSource None;  
[VB] Public Const None As UpdateRowSource  
[JScript] public var None : UpdateRowSource;
```

*Description*

Any returned parameters or rows are ignored.

*ToString*

```
[C#] public const UpdateRowSource OutputParameters;  
[C++] public: const UpdateRowSource OutputParameters;
```

1 *[VB] Public Const OutputParameters As UpdateRowSource*

2 *[JScript] public var OutputParameters : UpdateRowSource;*

3  
4 *Description*

5 *Output parameters are mapped to the changed row in the*

6 *System.Data.DataSet .*

7 *UpdateStatus enumeration (System.Data)*

8 *ToString*

9  
10  
11 *Description*

12 *Specifies the action to take with regard to the current and remaining rows*  
13 *during an System.Data.Common.DbDataAdapter.Update(System.Data.DataSet) .*

14 *ToString*

15  
16 *[C#] public const UpdateStatus Continue;*

17 *[C++] public: const UpdateStatus Continue;*

18 *[VB] Public Const Continue As UpdateStatus*

19 *[JScript] public var Continue : UpdateStatus;*

20  
21 *Description*

22 *The System.Data.Common.DataAdapter is to continue proccessing rows.*

23 *ToString*

24  
25 *[C#] public const UpdateStatus ErrorsOccurred;*

1 *[C++] public: const UpdateStatus ErrorsOccurred;*

2 *[VB] Public Const ErrorsOccurred As UpdateStatus*

3 *[JScript] public var ErrorsOccurred : UpdateStatus;*

4  
5 *Description*

6 *The event handler reports that the update should be treated as an error.*

7 *ToString*

8  
9 *[C#] public const UpdateStatus SkipAllRemainingRows;*

10 *[C++] public: const UpdateStatus SkipAllRemainingRows;*

11 *[VB] Public Const SkipAllRemainingRows As UpdateStatus*

12 *[JScript] public var SkipAllRemainingRows : UpdateStatus;*

13  
14 *Description*

15 *The current row and all remaining rows are not to be updated.*

16 *ToString*

17  
18 *[C#] public const UpdateStatus SkipCurrentRow;*

19 *[C++] public: const UpdateStatus SkipCurrentRow;*

20 *[VB] Public Const SkipCurrentRow As UpdateStatus*

21 *[JScript] public var SkipCurrentRow : UpdateStatus;*

22  
23 *Description*

24 *The current row is not to be updated.*

25 *VersionNotFoundException class (System.Data)*

*ToString*

*Description*

*Represents the exception that is thrown when attempting to return a version of a **System.Data.DataRow** that has been deleted.*

*VersionNotFoundException*

*Example Syntax:*

*ToString*

*[C#] public VersionNotFoundException();*

*[C++] public: VersionNotFoundException();*

*[VB] Public Sub New()*

*[JScript] public function VersionNotFoundException();*

*Description*

*Initializes a new instance of the **System.Data.VersionNotFoundException** class.*

*VersionNotFoundException*

*Example Syntax:*

*ToString*

*[C#] public VersionNotFoundException(string s);*

*[C++] public: VersionNotFoundException(String\* s);*

*[VB] Public Sub New(ByVal s As String)*

1 *[JScript] public function VersionNotFoundException(s : String);*

3 *Description*

4 *Initializes a new instance of the **System.Data.VersionNotFoundException***  
5 *class with the specified string. The string to display when the exception is thrown.*

6 *VersionNotFoundException*

7 *Example Syntax:*

8 *ToString*

10 *[C#] public VersionNotFoundException(SerializationInfo info, StreamingContext*  
11 *context);*

12 *[C++] public: VersionNotFoundException(SerializationInfo\* info,*  
13 *StreamingContext context);*

14 *[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As*  
15 *StreamingContext)*

16 *[JScript] public function VersionNotFoundException(info : SerializationInfo,*  
17 *context : StreamingContext);* *Initializes a new instance of the*  
18 ***System.Data.VersionNotFoundException** class.*

20 *Description*

21 *Initializes a new instance of the **System.Data.VersionNotFoundException***  
22 *class with serialization information. The data necessary to serialize or deserialize*  
23 *an object. Description of the source and destination of the specified serialized*  
24 *stream.*

25 *HelpLink*

1 *HResult*  
2 *InnerException*  
3 *Message*  
4 *Source*  
5 *StackTrace*  
6 *TargetSite*  
7 *XmlReadMode* enumeration (*System.Data*)  
8 *ToString*

9  
10  
11 *Description*

12 *Specifies how to read XML data and a relational schema into a*  
13 ***System.Data.DataSet*** .

14 *Use the members of this enumeration when setting the ReadMode*  
15 *parameter of the ***System.Data.DataSet.ReadXml(System.Xml.XmlReader)****  
16 *method.*

17 *ToString*

18  
19 *[C#] public const XmlReadMode Auto;*

20 *[C++] public: const XmlReadMode Auto;*

21 *[VB] Public Const Auto As XmlReadMode*

22 *[JScript] public var Auto : XmlReadMode;*

23  
24 *Description*  
25

1       *Default. Performs the most appropriate of these actions: If the data is a*  
2       *DiffGram, sets XmlReadMode to **DiffGram** .*

3       *ToString*

4  
5       [C#] *public const XmlReadMode DiffGram;*

6       [C++] *public: const XmlReadMode DiffGram;*

7       [VB] *Public Const DiffGram As XmlReadMode*

8       [JScript] *public var DiffGram : XmlReadMode;*

9  
10      *Description*

11       *Reads a DiffGram, applying changes from the DiffGram to the*  
12       ***System.Data.DataSet** . The semantics are identical to those of a*  
13       ***System.Data.DataSet.Merge(System.Data.DataSet)** operation. As with the*  
14       ***System.Data.DataSet.Merge(System.Data.DataSet)** operation,*  
15       ***System.Data.DataRow.RowState** values are preserved. Input to*  
16       ***System.Data.DataSet.ReadXml(System.Xml.XmlReader)** with DiffGrams should*  
17       *only be obtained using the output from*  
18       ***System.Data.DataSet.WriteXml(System.IO.Stream)** as a DiffGram.*

19      *ToString*

20  
21      [C#] *public const XmlReadMode Fragment;*

22      [C++] *public: const XmlReadMode Fragment;*

23      [VB] *Public Const Fragment As XmlReadMode*

24      [JScript] *public var Fragment : XmlReadMode;*

## Description

Reads XML documents containing inline XDR schema fragments, such as those generated by executing FOR XML schemas that include inline XDR schema against an instance of SQL Server. When **System.Data.XmlReadMode** is set to **SqlXml**, the default namespace is read as the inline schema.

## ToString

[C#] public const XmlReadMode IgnoreSchema;

[C++] public: const XmlReadMode IgnoreSchema;

[VB] Public Const IgnoreSchema As XmlReadMode

[JScript] public var IgnoreSchema : XmlReadMode;

## Description

Ignores any inline schema and reads data into the existing **System.Data.DataSet** schema. If any data does not match the existing schema, it is discarded (including data from differing namespaces defined for the **System.Data.DataSet**). If the data is a **DiffGram**, **IgnoreSchema** has the same functionality as **DiffGram**.

## ToString

[C#] public const XmlReadMode InferSchema;

[C++] public: const XmlReadMode InferSchema;

[VB] Public Const InferSchema As XmlReadMode

[JScript] public var InferSchema : XmlReadMode;

## *Description*

*Ignores any inline schema, inferring schema from the data, and loads the data. If the **System.Data.DataSet** already contains a schema, the current schema is extended by adding columns to existing tables, where they exist, and new tables where existing tables don't exist. An exception is thrown if a column already exists but has an incompatible mapping type property.*

## *ToString*

```
[C#] public const XmlReadMode ReadSchema;  
[C++] public: const XmlReadMode ReadSchema;  
[VB] Public Const ReadSchema As XmlReadMode  
[JScript] public var ReadSchema : XmlReadMode;
```

## *Description*

*Reads any inline schema and loads the data. If the **System.Data.DataSet** already contains schema, new tables may be added to the schema, but an exception is thrown if any tables in the inline schema already exist in the **System.Data.DataSet** .*

*XmlWriteMode enumeration (System.Data)*

## *ToString*

## *Description*

1       *Specifies how to write XML data and a relational schema from a*  
2 ***System.Data.DataSet*** .

3       *Use the members of this enumeration when setting the WriteMode*  
4 *parameter of the System.Data.DataSet.WriteXml(System.IO.Stream) method.*

5       *ToString*

6  
7 *[C#] public const XmlWriteMode DiffGram;*

8 *[C++] public: const XmlWriteMode DiffGram;*

9 *[VB] Public Const DiffGram As XmlWriteMode*

10 *[JScript] public var DiffGram : XmlWriteMode;*

11  
12 *Description*

13       *Writes the entire System.Data.DataSet as a DiffGram, including original*  
14 *and current values. To generate a DiffGram containing only changed values, call*  
15 ***System.Data.DataSet.GetChanges*** , and then call  
16 ***System.Data.DataSet.WriteXml(System.IO.Stream)*** as a DiffGram on the  
17 *returned System.Data.DataSet .*

18       *ToString*

19  
20 *[C#] public const XmlWriteMode IgnoreSchema;*

21 *[C++] public: const XmlWriteMode IgnoreSchema;*

22 *[VB] Public Const IgnoreSchema As XmlWriteMode*

23 *[JScript] public var IgnoreSchema : XmlWriteMode;*

24  
25 *Description*

1        *Writes the current contents of the **System.Data.DataSet** as XML data,*  
2        *without an XSD schema. If no data is loaded into the **System.Data.DataSet** ,*

## 6        **System.Data.Common**

### 8        *Description*

9        The **System.Data.Common** namespace contains classes shared by the  
10       .NET data providers.

11       DataAdapter class (**System.Data.Common**)

### 13       *Description*

14       Represents a set of data commands and a database connection that are used  
15       to fill the **System.Data.DataSet** and update the data source.

16       The **System.Data.Common.DataAdapter** serves as a bridge between a  
17       **System.Data.DataSet** and a data source for retrieving and saving data. The  
18       **System.Data.Common.DataAdapter** provides this bridge by mapping  
19       **System.Data.Common.DataAdapter.Fill(System.Data.DataSet)** , which  
20       changes the data in the **System.Data.DataSet** to match the data in the data source,  
21       and **System.Data.IDataAdapter.Update(System.Data.DataSet)** , which changes  
22       the data in the data source to match the data in the **System.Data.DataSet** .

23       Constructors:

24       DataAdapter

25       *Example Syntax:*

```

1
2 [C#] protected DataAdapter();
3 [C++] protected: DataAdapter();
4 [VB] Protected Sub New()
5 [JScript] protected function DataAdapter();
6

```

### 7 *Description*

8 Initializes a new instance of the **System.Data.Common.DataAdapter**  
9 class.

10 When an instance of **System.Data.Common.DataAdapter** is created, the  
11 following read/write properties are set to the following initial values.

12 Properties:

13 AcceptChangesDuringFill

```

14
15 [C#] public bool AcceptChangesDuringFill {get; set;}
16 [C++] public: __property bool get_AcceptChangesDuringFill();public: __property
17 void set_AcceptChangesDuringFill(bool);
18 [VB] Public Property AcceptChangesDuringFill As Boolean
19 [JScript] public function get AcceptChangesDuringFill() : Boolean;public function
20 set AcceptChangesDuringFill(Boolean);
21

```

### 22 *Description*

23 Gets or sets a value indicating whether  
24 **System.Data.DataRow.AcceptChanges** is called on a **System.Data.DataRow**  
25 after it is added to the **System.Data.DataTable** .

1 If false, **System.Data.DataRow.AcceptChanges** is not called, and the  
2 newly added rows are treated as inserted rows.

3 Container

4 DesignMode

5 Events

6 MissingMappingAction

7  
8  
9 *Description*

10 Determines the action to take when incoming data does not have a  
11 matching table or column.

12 The **System.Data.Common.DataAdapter.TableMappings** property  
13 provides the master mapping between the returned records and the  
14 **System.Data.DataSet**.

15 MissingSchemaAction

16  
17 [C#] public MissingSchemaAction MissingSchemaAction {get; set;}

18 [C++] public: \_\_property MissingSchemaAction

19 get\_MissingSchemaAction();public: \_\_property void

20 set\_MissingSchemaAction(MissingSchemaAction);

21 [VB] Public Property MissingSchemaAction As MissingSchemaAction

22 [JScript] public function get MissingSchemaAction() :

23 MissingSchemaAction;public function set

24 MissingSchemaAction(MissingSchemaAction);  
25

*Description*

Determines the action to take when existing **System.Data.DataSet** schema does not match incoming data.

Site

TableMappings

*Description*

Gets a collection that provides the master mapping between a source table and a **System.Data.DataTable** .

When reconciling changes, the **System.Data.Common.DataAdapter** uses the **System.Data.Common.DataTableMappingCollection** collection to associate the column names used by the data source with the column names used by the **System.Data.DataSet** .

Methods:

CloneInternals

[C#]	protected	virtual	DataAdapter	CloneInternals();
[C++]	protected:	virtual	DataAdapter*	CloneInternals();
[VB]	Overridable	Protected	Function CloneInternals()	As DataAdapter
[JScript]	protected	function	CloneInternals()	: DataAdapter;

*Description*

Creates a copy of this instance of **System.Data.Common.DataAdapter** .

*Return Value:* The cloned instance of **System.Data.Common.DataAdapter** .

All the commands, the **System.Data.Common.DataAdapter.TableMappings** , The **System.Data.Common.DataAdapter.MissingSchemaAction** , and the **System.Data.Common.DataAdapter.MissingMappingAction** are cloned. However, the connections for the commands are not copied, but shared. Thus, the cloned **System.Data.Common.DataAdapter** can be used against the same connection as the original.

#### CreateTableMappings

[C#] protected virtual DataTableMappingCollection CreateTableMappings();

[C++] protected: virtual DataTableMappingCollection\* CreateTableMappings();

[VB] Overridable Protected Function CreateTableMappings() As DataTableMappingCollection

[JScript] protected function CreateTableMappings() : DataTableMappingCollection;

#### Description

Creates a new **System.Data.Common.DataTableMappingCollection** .

*Return Value:* A new **System.Data.Common.DataTableMappingCollection** .

#### Dispose

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

1 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)  
 2 [JScript] protected override function Dispose(disposing : Boolean); Releases the  
 3 resources used by the **System.Data.Common.DataAdapter** .  
 4

5 *Description*

6 Releases the unmanaged resources used by the  
 7 **System.Data.Common.DataAdapter** and optionally releases the managed  
 8 resources.

9 This method is called by the public method and the  
 10 **System.Object.Finalize** method. **true** to release both managed and unmanaged  
 11 resources; **false** to release only unmanaged resources.

12 **Fill**

13  
 14 [C#] public abstract int Fill(DataSet dataSet);  
 15 [C++] public: virtual int Fill(DataSet\* dataSet) = 0;  
 16 [VB] MustOverride Public Function Fill(ByVal dataSet As DataSet) As Integer  
 17 [JScript] public abstract function Fill(dataSet : DataSet) : int;  
 18

19 *Description*

20 Adds or refreshes rows in the **System.Data.DataSet** to match those in the  
 21 data source using the **System.Data.DataSet** name, and creates a  
 22 **System.Data.DataTable** named "Table".

23 *Return Value:* The number of rows successfully added to or refreshed in the  
 24 **System.Data.DataSet** . This does not include rows affected by statements that do  
 25 not return rows.

The **System.Data.Common.DataAdapter.Fill(System.Data.DataSet)** method retrieves rows from the data source using the SELECT statement specified by an associated **System.Data.IDbDataAdapter.SelectCommand** property. The connection object associated with the SELECT statement must be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DataAdapter.Fill(System.Data.DataSet)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DataAdapter.Fill(System.Data.DataSet)** is called, it remains open. A **System.Data.DataSet** to fill with records and, if necessary, schema.

#### FillSchema

[C#] public abstract DataTable[] FillSchema(DataSet dataSet, SchemaType schemaType);

[C++] public: virtual DataTable\* FillSchema(DataSet\* dataSet, SchemaType schemaType) [] = 0;

[VB] MustOverride Public Function FillSchema(ByVal dataSet As DataSet, ByVal schemaType As SchemaType) As DataTable()

[JScript] public abstract function FillSchema(dataSet : DataSet, schemaType : SchemaType) : DataTable[];

#### Description

Adds a **System.Data.DataTable** named "Table" to the specified **System.Data.DataSet** and configures the schema to match that in the data source based on the specified **System.Data.SchemaType**.

*Return Value:* An array of **System.Data.DataTable** objects that contain schema information returned from the data source.

The **System.Data.Common.DataAdapter.FillSchema(System.Data.DataSet, System.Data.SchemaType)** method retrieves the schema from the data source using the **System.Data.IDbDataAdapter.SelectCommand** . The connection object associated with the **System.Data.IDbDataAdapter.SelectCommand** must be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DataAdapter.FillSchema(System.Data.DataSet, System.Data.SchemaType)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DataAdapter.FillSchema(System.Data.DataSet, System.Data.SchemaType)** is called, it remains open. The **System.Data.DataSet** to be filled with the schema from the data source. One of the **System.Data.SchemaType** values.

#### GetFillParameters

```
[C#]      public      abstract      IDataParameter[]      GetFillParameters();  
[C++]     public:     virtual      IDataParameter*      GetFillParameters()      []      =      0;  
[VB]      MustOverride Public Function GetFillParameters() As IDataParameter()  
[JScript] public abstract function GetFillParameters() : IDataParameter[];
```

#### *Description*

Gets the parameters set by the user when executing an SQL SELECT statement.

*Return Value:* An array of **System.Data.IDataParameter** objects that contains the parameters set by the user.

### ShouldSerializeTableMappings

```
[C#]      protected      virtual      bool      ShouldSerializeTableMappings();  
[C++]     protected:     virtual      bool      ShouldSerializeTableMappings();  
[VB]      Overridable    Protected    Function  ShouldSerializeTableMappings() As  
Boolean  
[JScript] protected      function  ShouldSerializeTableMappings() : Boolean;
```

### *Description*

Determines whether one or more **System.Data.Common.DataTableMapping** objects exist and they should be persisted.

*Return Value:* **true** if one or more **System.Data.Common.DataTableMapping** objects exist; otherwise **false**.

### Update

```
[C#]      public      abstract      int      Update(DataSet      dataSet);  
[C++]     public:     virtual      int      Update(DataSet*      dataSet)      =      0;  
[VB]      MustOverride Public Function Update(ByVal dataSet As DataSet) As  
Integer  
[JScript] public      abstract      function  Update(dataSet : DataSet) : int;
```

### *Description*

1 Calls the respective INSERT, UPDATE, or DELETE statements for each  
2 inserted, updated, or deleted row in the specified **System.Data.DataSet** from a  
3 **System.Data.DataTable** named "Table".

4 *Return Value:* The number of rows successfully updated from the  
5 **System.Data.DataSet**.

6 When an application calls the  
7 **System.Data.Common.DataAdapter.Update(System.Data.DataSet)** method,  
8 the **System.Data.Common.DataAdapter** examines the  
9 **System.Data.DataRow.RowState** property, and executes the required INSERT,  
10 UPDATE, or DELETE statements based on the order of the indexes configured in  
11 the **System.Data.DataSet**. For example,  
12 **System.Data.Common.DataAdapter.Update(System.Data.DataSet)** might  
13 execute a DELETE statement, followed by an INSERT statement, and then  
14 another DELETE statement, due to the ordering of the rows in the  
15 **System.Data.DataTable**. An application can call the  
16 **System.Data.DataSet.GetChanges** method in situations where you must control  
17 the sequence of statement types (for example, INSERTs before UPDATEs). For  
18 more information, see . The **System.Data.DataSet** used to update the data source.

19 DataColumnMapping class (System.Data.Common)

20 Update

### 23 *Description*

24 Contains a generic column mapping for an object that inherits from  
25 **System.Data.Common.DataAdapter**. This class cannot be inherited.

A **System.Data.Common.DataColumnMapping** enables you to use column names in a **System.Data.DataTable** that are different from those in the data source. The **DataAdapter** uses the mapping to match the columns when the tables in the **System.Data.DataSet** or data source are updated. For more information, see .

**DataColumnMapping**

*Example Syntax:*

Update

```
[C#]          public          DataColumnMapping();
[C++]         public:         DataColumnMapping();
[VB]          Public          Sub          New()
[JavaScript] public function DataColumnMapping(); Initializes a new instance of the
System.Data.Common.DataColumnMapping class.
```

*Description*

Initializes a new instance of the **System.Data.Common.DataColumnMapping** class.

**DataColumnMapping**

*Example Syntax:*

Update

```
[C#] public DataColumnMapping(string sourceColumn, string dataSetColumn);
[C++] public: DataColumnMapping(String* sourceColumn, String*
dataSetColumn);
```

1 [VB] Public Sub New(ByVal sourceColumn As String, ByVal dataSetColumn As  
2 String)

3 [JScript] public function DataColumnMapping(sourceColumn : String,  
4 dataSetColumn : String);

5  
6 *Description*

7 Initializes a new instance of the  
8 **System.Data.Common.DataColumnMapping** class when given a source column  
9 name and a **System.Data.DataSet** column name to map to. The case-sensitive  
10 column name from a data source. The column name, which is not case sensitive,  
11 from a **System.Data.DataSet** to map to.

12 DataSetColumn

13 Update

14  
15 [C#] public string DataSetColumn {get; set;}

16 [C++] public: \_\_property String\* get\_DataSetColumn();public: \_\_property void  
17 set\_DataSetColumn(String\*);

18 [VB] Public Property DataSetColumn As String

19 [JScript] public function get DataSetColumn() : String;public function set  
20 DataSetColumn(String);

21  
22 *Description*

23 Gets or sets the name of the column within the **System.Data.DataSet** to  
24 map to.

25 SourceColumn

## Update

[C#]        public        string        SourceColumn        {get;        set;}

[C++] public: \_\_property String\* get\_SourceColumn();public: \_\_property void  
set\_SourceColumn(String\*);

[VB]        Public        Property        SourceColumn        As        String

[JScript] public function get SourceColumn() : String;public function set  
SourceColumn(String);

### *Description*

Gets or sets the case-sensitive column name from a data source to map  
from.

### GetDataColumnBySchemaAction

[C#] public DataColumn GetDataColumnBySchemaAction(DataTable dataTable,  
Type        dataType,        MissingSchemaAction        schemaAction);

[C++] public: DataColumn\* GetDataColumnBySchemaAction(DataTable\*  
dataTable,    Type\*    dataType,    MissingSchemaAction    schemaAction);

[VB] Public Function GetDataColumnBySchemaAction(ByVal dataTable As  
DataTable, ByVal dataType As Type, ByVal schemaAction As  
MissingSchemaAction)        As        DataColumn

[JScript] public function GetDataColumnBySchemaAction(dataTable : DataTable,  
dataType : Type, schemaAction : MissingSchemaAction) : DataColumn;

### *Description*

Gets a **System.Data.DataColumn** from the given **System.Data.DataTable** using the **System.Data.MissingSchemaAction** and the **System.Data.Common.DataColumnMapping.DataSetColumn** property.

*Return Value:* A **System.Data.DataColumn**.

If the given *dataType* is not convertible to the **System.Type** of the **System.Data.DataColumn**, an exception is generated. The **System.Data.DataTable** to get the column from. The **System.Type** of the data column. One of the **System.Data.MissingSchemaAction** values.

**ICloneable.Clone**

[C#] object ICloneable.Clone();

[C++] Object\* ICloneable::Clone();

[VB] Function Clone() As Object Implements ICloneable.Clone

[JScript] function ICloneable.Clone() : Object;

**ToString**

[C#] public override string ToString();

[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

*Description*

Converts the current **System.Data.Common.DataColumnMapping.SourceColumn** name to a string.



```

1  [C++]      public:      __property      int      get_Count();
2  [VB]      Public      ReadOnly      Property      Count      As      Integer
3  [JScript]      public      function      get      Count()      :      int;

```

5 *Description*

6 Gets the number of items in the collection.

7 Item

8 ToString

```

10 [C#]      public      DataColumnMapping      this[int      index]      {get;      set;}

```

```

11 [C++]      public:      __property      DataColumnMapping*      get_Item(int      index);public:
12 __property      void      set_Item(int      index,      DataColumnMapping*);

```

```

13 [VB]      Public      Default      Property      Item(ByVal      index      As      Integer)      As
14 DataColumnMapping

```

```

15 [JScript]      returnValue      =
16 DataColumnMappingCollectionObject.Item(index);DataColumnMappingCollectio
17 nObject.Item(index)      =      returnValue;      Gets      or      sets      the
18 System.Data.Common.DataColumnMapping      object      specified.

```

20 *Description*

21 Gets or sets the **System.Data.Common.DataColumnMapping** object at  
22 the specified index. The zero-based index of the  
23 **System.Data.Common.DataColumnMapping** object to find.

24 Item

25 ToString

```

1
2 [C#] public DataColumnMapping this[string sourceColumn] {get; set;}
3 [C++] public: __property DataColumnMapping* get_Item(String*
4 sourceColumn);public: __property void set_Item(String* sourceColumn,
5 DataColumnMapping*);
6 [VB] Public Default Property Item(ByVal sourceColumn As String) As
7 DataColumnMapping
8 [JScript] return Value =
9 DataColumnMappingCollectionObject.Item(sourceColumn);DataColumnMapping
10 CollectionObject.Item(sourceColumn) = return Value;
11

```

## 12 *Description*

13 Gets or sets the **System.Data.Common.DataColumnMapping** object with  
14 the specified source column name. The case-sensitive name of the source column.

## 15 *Add*

```

16
17 [C#] public int Add(object value);
18 [C++] public: __sealed int Add(Object* value);
19 [VB] NotOverridable Public Function Add(ByVal value As Object) As Integer
20 [JScript] public function Add(value : Object) : int; Adds a column mapping to the
21 collection.
22

```

## 23 *Description*

24

25

1 Adds an **System.Object** to the collection.

2 *Return Value:* The index of the **System.Object** added to the collection. An  
3 **System.Object** to add to the collection.

4 Add

5  
6 [C#] public DataColumnMapping Add(string sourceColumn, string  
7 dataSetColumn);

8 [C++] public: DataColumnMapping\* Add(String\* sourceColumn, String\*  
9 dataSetColumn);

10 [VB] Public Function Add(ByVal sourceColumn As String, ByVal  
11 dataSetColumn As String) As DataColumnMapping

12 [JScript] public function Add(sourceColumn : String, dataSetColumn : String) :  
13 DataColumnMapping;

14  
15 *Description*

16 Adds a column mapping to the collection when given a source column  
17 name and a **System.Data.DataSet** column name.

18 *Return Value:* The **System.Data.Common.DataColumnMapping** object added  
19 to the collection. The case-sensitive name of the source column to map to. The  
20 name, which is not case sensitive, of the **System.Data.DataSet** column to map to.

21 AddRange

22  
23 [C#] public void AddRange(DataColumnMapping[] values);

24 [C++] public: void AddRange(DataColumnMapping\* values[]);

25 [VB] Public Sub AddRange(ByVal values() As DataColumnMapping)



1 **System.Object** exists in the collection.  
 2 *Return Value:* **true** if the collection contains the specified  
 3 **System.Data.Common.DataColumnMapping** object; otherwise, **false** . An  
 4 **System.Object** that is the **System.Data.Common.DataColumnMapping**.

5 Contains

6  
 7 [C#] public bool Contains(string value);  
 8 [C++] public: \_\_sealed bool Contains(String\* value);  
 9 [VB] NotOverridable Public Function Contains(ByVal value As String) As  
 10 Boolean  
 11 [JScript] public function Contains(value : String) : Boolean; Gets a value  
 12 indicating whether a **System.Data.Common.DataColumnMapping** object exists  
 13 in the collection.

14  
 15 *Description*

16 Gets a value indicating whether a  
 17 **System.Data.Common.DataColumnMapping** object with the given value exists  
 18 in the collection.

19 *Return Value:* **true** if collection contains a  
 20 **System.Data.Common.DataColumnMapping** object with this source column  
 21 name; otherwise, **false** . The case-sensitive source column name of the  
 22 **System.Data.Common.DataColumnMapping** object.

23 CopyTo

24  
 25 [C#] public void CopyTo(Array array, int index);

```

1 [C++] public: __sealed void CopyTo(Array* array, int index);
2 [VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As
3 Integer)
4 [JScript] public function CopyTo(array : Array, index : int);
5

```

#### 6 *Description*

7 Copies the elements of the  
8 **System.Data.Common.DataColumnMappingCollection** to the specified array.  
9 An **System.Array** to which to copy  
10 **System.Data.Common.DataColumnMappingCollection** elements. The starting  
11 index of the array.

#### 12 **GetByDataSetColumn**

```

13
14 [C#] public DataColumnMapping GetByDataSetColumn(string value);
15 [C++] public: DataColumnMapping* GetByDataSetColumn(String* value);
16 [VB] Public Function GetByDataSetColumn(ByVal value As String) As
17 DataColumnMapping
18 [JScript] public function GetByDataSetColumn(value : String) :
19 DataColumnMapping;
20

```

#### 21 *Description*

22 Gets the **System.Data.Common.DataColumnMapping** object with the  
23 specified **System.Data.DataSet** column name.  
24 *Return Value:* The **System.Data.Common.DataColumnMapping** object with the  
25

specified **System.Data.DataSet** column name. The name, which is not case-sensitive, of the **System.Data.DataSet** column to find.

**GetColumnMappingBySchemaAction**

[C#]                    public                    static                    DataColumnMapping  
GetColumnMappingBySchemaAction(DataColumnMappingCollection  
columnMappings, string sourceColumn, MissingMappingAction mappingAction);

[C++]                    public:                    static                    DataColumnMapping\*  
GetColumnMappingBySchemaAction(DataColumnMappingCollection\*  
columnMappings,        String\*        sourceColumn,        MissingMappingAction  
mappingAction);

[VB] Public Shared Function GetColumnMappingBySchemaAction(ByVal  
columnMappings As DataColumnMappingCollection, ByVal sourceColumn As  
String, ByVal mappingAction As MissingMappingAction) As  
DataColumnMapping

[JScript]                    public                    static                    function  
GetColumnMappingBySchemaAction(columnMappings :  
DataColumnMappingCollection, sourceColumn : String, mappingAction :  
MissingMappingAction) :                    DataColumnMapping;

### *Description*

Gets a **System.Data.Common.DataColumnMapping** for the specified  
**System.Data.Common.DataColumnMappingCollection** , source column name,  
and **System.Data.MissingMappingAction** .

*Return Value:* A **System.Data.Common.DataColumnMapping** object.

If the **System.Data.Common.DataColumnMapping** exists in the collection, it is returned. the **System.Data.Common.DataColumnMappingCollection**. The case-sensitive source column name to find. One of the **System.Data.MissingMappingAction** values.

#### GetEnumerator

[C#] public IEnumerator GetEnumerator();  
 [C++] public: \_\_sealed IEnumerator\* GetEnumerator();  
 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator  
 [JScript] public function GetEnumerator() : IEnumerator;

#### Description

#### IndexOf

[C#] public int IndexOf(object value);  
 [C++] public: \_\_sealed int IndexOf(Object\* value);  
 [VB] NotOverridable Public Function IndexOf(ByVal value As Object) As Integer  
 [JScript] public function IndexOf(value : Object) : int; Gets the location of the specified **System.Data.Common.DataColumnMapping** within the collection.

#### Description

Gets the location of the specified **System.Object** that is a **System.Data.Common.DataColumnMapping** within the collection.

*Return Value:* The location of the specified **System.Object** that is a

**System.Data.Common.DataColumnMapping** within the collection. An **System.Object** that is the **System.Data.Common.DataColumnMapping** to find.

**IndexOf**

```
[C#]      public      int      IndexOf(string      sourceColumn);
[C++]     public:      __sealed  int      IndexOf(String*      sourceColumn);
[VB] NotOverridable Public Function IndexOf(ByVal sourceColumn As String)
As
Integer
[JScript] public  function  IndexOf(sourceColumn : String) : int;
```

#### *Description*

Gets the location of the **System.Data.Common.DataColumnMapping** with the specified source column name.

**Return Value:** The location of the **System.Data.Common.DataColumnMapping** with the specified case-sensitive source column name. The case-sensitive name of the source column.

**IndexOfDataSetColumn**

```
[C#]      public      int      IndexOfDataSetColumn(string      dataSetColumn);
[C++]     public:      int      IndexOfDataSetColumn(String*      dataSetColumn);
[VB] Public Function IndexOfDataSetColumn(ByVal dataSetColumn As String)
As
Integer
[JScript] public  function  IndexOfDataSetColumn(dataSetColumn : String) : int;
```

#### *Description*

Gets the location of the specified **System.Data.Common.DataColumnMapping** with the given **System.Data.DataSet** column name.

*Return Value:* The location of the specified **System.Data.Common.DataColumnMapping** with the given data set column name, or -1 if the **System.Data.Common.DataColumnMapping** object does not exist in the collection. The name, which is not case-sensitive, of the data set column to find.

#### Insert

[C#] public void Insert(int index, object value);

[C++] public: \_\_sealed void Insert(int index, Object\* value);

[VB] NotOverridable Public Sub Insert(ByVal index As Integer, ByVal value As Object)

[JScript] public function Insert(index : int, value : Object);

#### Description

Inserts a **System.Data.Common.DataColumnMapping** object into the **System.Data.Common.DataColumnMappingCollection** at the specified index.

*Return Value:* A **System.Data.Common.DataColumnMapping** object. The zero-based index of the **System.Data.Common.DataColumnMapping** object to insert. The **System.Data.Common.DataColumnMapping** object.

#### Remove

[C#] public void Remove(object value);

1 [C++] public: \_\_sealed void Remove(Object\* value);  
 2 [VB] NotOverridable Public Sub Remove(ByVal value As Object)  
 3 [JScript] public function Remove(value : Object);  
 4

#### 5 *Description*

6 Removes the **System.Object** that is a  
 7 **System.Data.Common.DataColumnMapping** from the collection. The  
 8 **System.Object** that is the **System.Data.Common.DataColumnMapping** to  
 9 remove.

#### 10 RemoveAt

11  
 12 [C#] public void RemoveAt(int index);  
 13 [C++] public: \_\_sealed void RemoveAt(int index);  
 14 [VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)  
 15 [JScript] public function RemoveAt(index : int); Removes the specified  
 16 **System.Data.Common.DataColumnMapping** object from the collection.  
 17

#### 18 *Description*

19 Removes the **System.Data.Common.DataColumnMapping** object with  
 20 the specified index from the collection. The zero-based index of the  
 21 **System.Data.Common.DataColumnMapping** object to remove.

#### 22 RemoveAt

23  
 24 [C#] public void RemoveAt(string sourceColumn);  
 25 [C++] public: \_\_sealed void RemoveAt(String\* sourceColumn);

```

1 [VB] NotOverridable Public Sub RemoveAt(ByVal sourceColumn As String)
2 [JScript]    public    function    RemoveAt(sourceColumn    :    String);

```

```

3
4 Description

```

```

5     Removes the System.Data.Common.DataColumnMapping object with
6 the specified source column name from the collection. The case-sensitive source
7 column name.

```

```

8     DataColumnMappingCollection.Add

```

```

9
10 [C#]          DataColumnMapping          DataColumnMappingCollection.Add(string
11 sourceColumnName,          string          dataSetColumnName);

```

```

12 [C++]          DataColumnMapping*          DataColumnMappingCollection::Add(String*
13 sourceColumnName,          String*          dataSetColumnName);

```

```

14 [VB]    Function    Add(ByVal    sourceColumnName    As    String,    ByVal
15 dataSetColumnName    As    String)    As    DataColumnMapping    Implements
16 DataColumnMappingCollection.Add

```

```

17 [JScript] function DataColumnMappingCollection.Add(sourceColumnName : String,
18 dataSetColumnName : String) : DataColumnMapping;

```

```

19     DataColumnMappingCollection.GetDataSetColumn

```

```

20
21 [C#] DataColumnMapping DataColumnMappingCollection.GetDataSetColumn(string
22 dataSetColumnName);

```

```

23 [C++]          DataColumnMapping*
24 DataColumnMappingCollection::GetDataSetColumn(String*
25 dataSetColumnName);

```

```

1 [VB] Function GetByDataSetColumn(ByVal dataSetColumnName As String) As
2 IColumnMapping Implements IColumnMappingCollection.GetByDataSetColumn
3 [JScript] function
4 IColumnMappingCollection.GetByDataSetColumn(dataSetColumnName : String)
5 : IColumnMapping;

```

```

6     DataTableMapping class (System.Data.Common)
7     ToString

```

10 *Description*

11 Contains a description of a mapped relationship between a source table and  
12 a **System.Data.DataTable** . This class is used by a  
13 **System.Data.Common.DataAdapter** when populating a **System.Data.DataSet** .

14 A **System.Data.Common.DataTableMapping** provides a master mapping  
15 between the data returned from a query against a data source, and a  
16 **System.Data.DataTable** . The **System.Data.Common.DataTableMapping**  
17 name can be passed in place of the **System.Data.DataTable** name to the **Fill**  
18 method of the **DataAdapter**. For more information, see .

19 DataTableMapping

20 *Example Syntax:*

21 ToString

```

23 [C#] public DataTableMapping();
24 [C++] public: DataTableMapping();
25 [VB] Public Sub New()

```

[JScript] public function DataTableMapping(); Initializes a new instance of the **System.Data.Common.DataTableMapping** class.

*Description*

Initializes a new instance of the **System.Data.Common.DataTableMapping** class.

DataTableMapping

*Example Syntax:*

ToString

[C#] public DataTableMapping(string sourceTable, string dataSetTable);

[C++] public: DataTableMapping(String\* sourceTable, String\* dataSetTable);

[VB] Public Sub New(ByVal sourceTable As String, ByVal dataSetTable As String)

[JScript] public function DataTableMapping(sourceTable : String, dataSetTable : String);

*Description*

Initializes a new instance of the **System.Data.Common.DataTableMapping** class with a source when given a source table name and a **System.Data.DataTable** name. The case-sensitive source table name from a data source. The table name from a **System.Data.DataSet** to map to.

DataTableMapping

*Example Syntax:*

ToString

```
[C#] public DataTableMapping(string sourceTable, string dataSetTable,
DataColumnMapping[] columnMappings);
[C++] public: DataTableMapping(String* sourceTable, String* dataSetTable,
DataColumnMapping* columnMappings[]);
[VB] Public Sub New(ByVal sourceTable As String, ByVal dataSetTable As
String, ByVal columnMappings() As DataColumnMapping)
[JavaScript] public function DataTableMapping(sourceTable : String, dataSetTable :
String, columnMappings : DataColumnMapping[]);
```

### *Description*

Initializes a new instance of the **System.Data.Common.DataTableMapping** class when given a source table name, a **System.Data.DataTable** name, and an array of **System.Data.Common.DataColumnMapping** objects. The case-sensitive source table name from a data source. The table name from a **System.Data.DataSet** to map to. An array of **System.Data.Common.DataColumnMapping** objects.

ColumnMappings

ToString

```
[C#] public DataColumnMappingCollection ColumnMappings {get;}
[C++] public: __property DataColumnMappingCollection*
get_ColumnMappings();
[VB] Public ReadOnly Property ColumnMappings As
```

```

1 DataColumnMappingCollection
2 [JScript]      public      function      get      ColumnMappings()      :
3 DataColumnMappingCollection;

```

5 *Description*

6 Gets the **System.Data.Common.DataColumnMappingCollection** for the  
7 **System.Data.DataTable** .

8 DataSetTable

9 ToString

11 [C#] public string DataSetTable {get; set;}

12 [C++] public: \_\_property String\* get\_DataSetTable();public: \_\_property void  
13 set\_DataSetTable(String\*);

14 [VB] Public Property DataSetTable As String

15 [JScript] public function get DataSetTable() : String;public function set  
16 DataSetTable(String);

18 *Description*

19 Gets or sets the table name from a **System.Data.DataSet** .

20 SourceTable

21 ToString

23 [C#] public string SourceTable {get; set;}

24 [C++] public: \_\_property String\* get\_SourceTable();public: \_\_property void  
25 set\_SourceTable(String\*);

```
1 [VB]      Public      Property      SourceTable      As      String
2 [JScript] public function get SourceTable() : String;public function set
3 SourceTable(String);
```

#### 5 *Description*

6 Gets or sets the case-sensitive source table name from a data source.

7 **GetColumnMappingBySchemaAction**

```
9 [C#] public DataColumnMapping GetColumnMappingBySchemaAction(string
10 sourceColumn,      MissingMappingAction      mappingAction);
```

```
11 [C++]      public:      DataColumnMapping*
12 GetColumnMappingBySchemaAction(String*      sourceColumn,
13 MissingMappingAction      mappingAction);
```

```
14 [VB]      Public      Function      GetColumnMappingBySchemaAction(ByVal
15 sourceColumn As String, ByVal mappingAction As MissingMappingAction) As
16 DataColumnMapping
```

```
17 [JScript] public function GetColumnMappingBySchemaAction(sourceColumn :
18 String, mappingAction : MissingMappingAction) : DataColumnMapping;
```

#### 20 *Description*

21 Gets a **System.Data.DataColumn** from the specified  
22 **System.Data.DataTable** using the specified  
23 **System.Data.MissingMappingAction** value and the name of the  
24 **System.Data.DataColumn**.

25 *Return Value:* A **System.Data.DataColumn**. The name of the

**System.Data.DataColumn** . One of the **System.Data.MissingMappingAction** values.

**GetDataTableBySchemaAction**

[C#] public DataTable GetDataTableBySchemaAction(DataSet dataSet, MissingSchemaAction schemaAction);

[C++] public: DataTable\* GetDataTableBySchemaAction(DataSet\* dataSet, MissingSchemaAction schemaAction);

[VB] Public Function GetDataTableBySchemaAction(ByVal dataSet As DataSet, ByVal schemaAction As MissingSchemaAction) As DataTable

[JScript] public function GetDataTableBySchemaAction(dataSet : DataSet, schemaAction : MissingSchemaAction) : DataTable;

#### *Description*

Gets the current **System.Data.DataTable** for the specified **System.Data.DataSet** using the specified **System.Data.MissingSchemaAction** value.

*Return Value:* A **System.Data.DataTable** .

If the **System.Data.DataTable** does not exist, the specified **System.Data.MissingSchemaAction** is taken. The **System.Data.DataSet** from which to get the **System.Data.DataTable** . One of the **System.Data.MissingSchemaAction** values.

**ICloneable.Clone**

[C#] object ICloneable.Clone();

```

1  [C++]          Object*          ICloneable::Clone();
2  [VB]   Function  Clone()  As  Object  Implements  ICloneable.Clone
3  [JScript] function ICloneable.Clone() : Object;
4          ToString
5
6  [C#]          public          override          string          ToString();
7  [C++]          public:          String*          ToString();
8  [VB]   Overrides  Public  Function  ToString()  As  String
9  [JScript]   public  override  function  ToString()  :  String;
10
11 Description
12
13   Converts          the          current
14   System.Data.Common.DataTableMapping.SourceTable name to a string.
15
16   Return Value:          The          current
17   System.Data.Common.DataTableMapping.SourceTable name, as a string.
18
19   DataTableMappingCollection class (System.Data.Common)
20   ToString
21
22 Description
23
24   A collection of System.Data.Common.DataTableMapping objects. This
25   class cannot be inherited.
26
27   DataTableMappingCollection
28
29   Example Syntax:
30
31   ToString

```

```

1
2 [C#] public DataTableMappingCollection();
3 [C++] public: DataTableMappingCollection();
4 [VB] Public Sub New()
5 [JScript] public function DataTableMappingCollection();
6

```

### Description

Initializes an empty

**System.Data.Common.DataTableMappingCollection .**

Count

ToString

```

13 [C#] public int Count {get;}
14 [C++] public: __property int get_Count();
15 [VB] Public ReadOnly Property Count As Integer
16 [JScript] public function get Count() : int;
17

```

### Description

Gets the number of items in the collection.

Item

ToString

```

23 [C#] public DataTableMapping this[int index] {get; set;}
24 [C++] public: __property DataTableMapping* get_Item(int index);public:
25 __property void set_Item(int index, DataTableMapping*);

```

```

1  [VB] Public Default Property Item(ByVal index As Integer) As
2  DataTableMapping
3  [JScript]                                     returnValue =
4  DataTableMappingCollectionObject.Item(index);DataTableMappingCollectionOb
5  ject.Item(index) = returnValue; Gets or sets the
6  System.Data.Common.DataTableMapping object specified.

```

### 8 *Description*

9 Gets or sets the **System.Data.Common.DataTableMapping** object at a  
10 specified index. The zero-based index of the  
11 **System.Data.Common.DataTableMapping** object to find.

12 Item

13 ToString

```

14
15 [C#] public DataTableMapping this[string sourceTable] {get; set;}
16 [C++] public: __property DataTableMapping* get_Item(String*
17 sourceTable);public: __property void set_Item(String* sourceTable,
18 DataTableMapping*);

```

```

19 [VB] Public Default Property Item(ByVal sourceTable As String) As
20 DataTableMapping
21 [JScript]                                     returnValue =
22 DataTableMappingCollectionObject.Item(sourceTable);DataTableMappingCollect
23 ionObject.Item(sourceTable) = returnValue;

```

### 25 *Description*



1 Adds a table mapping to the collection when given a source table name and  
 2 a **System.Data.DataSet** table name.  
 3 *Return Value:* The **System.Data.Common.DataTableMapping** object that was  
 4 added to the collection. The case-sensitive name of the source table to map to. The  
 5 name, which is not case-sensitive, of the **System.Data.DataSet** table to map to.

## 6 AddRange

7  
 8 [C#] public void AddRange(DataTableMapping[] values);  
 9 [C++] public: void AddRange(DataTableMapping\* values[]);  
 10 [VB] Public Sub AddRange(ByVal values() As DataTableMapping)  
 11 [JScript] public function AddRange(values : DataTableMapping[]);  
 12

## 13 Description

14 Copies the elements of the specified  
 15 **System.Data.Common.DataTableMapping** array to the end of the collection.

## 16 Clear

17  
 18 [C#] public void Clear();  
 19 [C++] public: \_\_sealed void Clear();  
 20 [VB] NotOverridable Public Sub Clear()  
 21 [JScript] public function Clear();  
 22

## 23 Description

24 Removes all items from the collection.

25 Contains

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]          public          bool          Contains(object          value);
[C++]         public:         __sealed      bool          Contains(Object*          value);
[VB] NotOverridable Public Function Contains(ByVal value As Object) As
Boolean
[JScript]     public  function  Contains(value : Object) : Boolean;
```

*Description*

Gets a value indicating whether the given **System.Data.Common.DataTableMapping** object exists in the collection.  
*Return Value:* **true** if this collection contains the specified **System.Data.Common.DataTableMapping** ; otherwise, **false** . An **System.Object** that is the **System.Data.Common.DataTableMapping**.

Contains

```
[C#]          public          bool          Contains(string          value);
[C++]         public:         __sealed      bool          Contains(String*          value);
[VB] NotOverridable Public Function Contains(ByVal value As String) As
Boolean
[JScript]     public  function  Contains(value : String) : Boolean; Gets a value
indicating whether a System.Data.Common.DataTableMapping object exists in
the collection.
```

*Description*

Gets a value indicating whether a **System.Data.Common.DataTableMapping** object with the given source table name exists in the collection.

*Return Value:* **true** if the collection contains a **System.Data.Common.DataTableMapping** object with this source table name; otherwise, **false**. The case-sensitive source table name containing the **System.Data.Common.DataTableMapping** object.

#### CopyTo

[C#] public void CopyTo(Array array, int index);

[C++] public: \_\_sealed void CopyTo(Array\* array, int index);

[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)

[JScript] public function CopyTo(array : Array, index : int);

#### Description

Copies the elements of the **System.Data.Common.DataTableMappingCollection** to the specified array. An **System.Array** to which to copy **System.Data.Common.DataTableMappingCollection** elements. The starting index of the array.

#### GetByDataSetTable

[C#] public DataTableMapping GetByDataSetTable(string dataSetTable);

[C++] public: DataTableMapping\* GetByDataSetTable(String\* dataSetTable);

```

1  [VB] Public Function GetByDataSetTable(ByVal dataSetTable As String) As
2  DataTableMapping
3  [JScript] public function GetByDataSetTable(dataSetTable : String) :
4  DataTableMapping;

```

### 6 *Description*

7 Gets the **System.Data.Common.DataTableMapping** object with the  
8 specified **System.Data.DataSet** table name.

9 *Return Value:* The **System.Data.Common.DataTableMapping** object with the  
10 specified **System.Data.DataSet** table name. The name, which is not case  
11 sensitive, of the **System.Data.DataSet** table to find.

### 12 *GetEnumerator*

```

13
14 [C#] public IEnumerator GetEnumerator();
15 [C++] public: __sealed IEnumerator* GetEnumerator();
16 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator
17 [JScript] public function GetEnumerator() : IEnumerator;

```

### 19 *Description*

#### 20 *GetTableMappingBySchemaAction*

```

21
22 [C#] public static DataTableMapping
23 GetTableMappingBySchemaAction(DataTableMappingCollection tableMappings,
24 string sourceTable, string dataSetTable, MissingMappingAction mappingAction);
25 [C++] public: static DataTableMapping*

```

```

1  GetTableMappingBySchemaAction(DataTableMappingCollection*
2  tableMappings,      String*      sourceTable,      String*      dataSetTable,
3  MissingMappingAction                                mappingAction);
4  [VB] Public Shared Function GetTableMappingBySchemaAction(ByVal
5  tableMappings As DataTableMappingCollection, ByVal sourceTable As String,
6  ByVal dataSetTable As String, ByVal mappingAction As MissingMappingAction)
7  As
8  [JScript] public static function GetTableMappingBySchemaAction(tableMappings
9  : DataTableMappingCollection, sourceTable : String, dataSetTable : String,
10 mappingAction : MissingMappingAction) : DataTableMapping;

```

## 12 *Description*

13 Gets a **System.Data.Common.DataColumnMapping** object with the  
14 given source table name and **System.Data.DataSet** table name, using the given  
15 **System.Data.MissingMappingAction**

16 *Return Value:* A **System.Data.Common.DataTableMapping** .

17 If the **System.Data.Common.DataTableMapping** exists in the collection,  
18 it is returned. The **System.Data.Common.DataTableMappingCollection**  
19 collection to search. The case-sensitive name of the source table to find. The  
20 name, which is not case-sensitive, to assign to the **System.Data.DataSet** table.  
21 One of the **System.Data.MissingMappingAction** values.

## 22 *Index Of*

```

23
24 [C#]      public      int      IndexOf(object      value);
25 [C++]     public:     __sealed  int      IndexOf(Object*  value);

```

[VB] NotOverridable Public Function IndexOf(ByVal value As Object) As Integer  
 [JScript] public function IndexOf(value : Object) : int; Gets the location of the specified **System.Data.Common.DataTableMapping** object within the collection.

#### *Description*

Gets the location of the specified **System.Object** that is a **System.Data.Common.DataTableMapping** object within the collection.

*Return Value:* The location of the specified **System.Object** that is a **System.Data.Common.DataTableMapping** object within the collection. An **System.Object** that is the **System.Data.Common.DataTableMapping** object to find.

#### **IndexOf**

[C#] public int IndexOf(string sourceTable);

[C++] public: \_\_sealed int IndexOf(String\* sourceTable);

[VB] NotOverridable Public Function IndexOf(ByVal sourceTable As String) As Integer

[JScript] public function IndexOf(sourceTable : String) : int;

#### *Description*

Gets the location of the **System.Data.Common.DataTableMapping** object with the specified source table name.

*Return Value:* The location of the **System.Data.Common.DataTableMapping**

object with the specified source table name. The case-sensitive name of the source table.

### IndexOfDataSetTable

[C#] public int IndexOfDataSetTable(string dataSetTable);

[C++] public: int IndexOfDataSetTable(String\* dataSetTable);

[VB] Public Function IndexOfDataSetTable(ByVal dataSetTable As String) As Integer

[JScript] public function IndexOfDataSetTable(dataSetTable : String) : int;

### *Description*

Gets the location of the **System.Data.Common.DataTableMapping** object with the specified **System.Data.DataSet** table name.

*Return Value:* The location of the **System.Data.Common.DataTableMapping** object with the given **System.Data.DataSet** table name, or -1 if the **System.Data.Common.DataTableMapping** object does not exist in the collection. The name, which is not case-sensitive, of the data set table to find.

### Insert

[C#] public void Insert(int index, object value);

[C++] public: \_\_sealed void Insert(int index, Object\* value);

[VB] NotOverridable Public Sub Insert(ByVal index As Integer, ByVal value As Object)

[JScript] public function Insert(index : int, value : Object);

## Description

Inserts a **System.Data.Common.DataTableMapping** object into the **System.Data.Common.DataTableMappingCollection** at the specified index.

*Return Value:* A **System.Data.Common.DataTableMapping** object. The zero-based index of the **System.Data.Common.DataTableMapping** object to insert. The **System.Data.Common.DataTableMapping** object.

## Remove

```
[C#]          public          void          Remove(object          value);
[C++]         public:         __sealed      void          Remove(Object*      value);
[VB] NotOverridable Public Sub Remove(ByVal value As Object)
[JScript]     public         function      Remove(value          :          Object);
```

## Description

Removes the specified **System.Data.Common.DataTableMapping** object from the collection. The **System.Object** that is the **System.Data.Common.DataTableMapping** object to remove.

## RemoveAt

```
[C#]          public          void          RemoveAt(int          index);
[C++]         public:         __sealed      void          RemoveAt(int          index);
[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)
[JScript]     public         function      RemoveAt(index : int); Removes the specified
System.Data.Common.DataTableMapping object from the collection.
```

## Description

Removes the **System.Data.Common.DataTableMapping** object located at the specified index from the collection. The zero-based index of the **System.Data.Common.DataTableMapping** object to remove.

### RemoveAt

```
[C#]      public      void      RemoveAt(string      sourceTable);
[C++]    public:    __sealed    void    RemoveAt(String*    sourceTable);
[VB]    NotOverridable Public Sub RemoveAt(ByVal sourceTable As String)
[JScript] public      function    RemoveAt(sourceTable      :      String);
```

## Description

Removes the **System.Data.Common.DataTableMapping** object with the specified source table name from the collection. The case-sensitive source table name to find.

### ITableMappingCollection.Add

```
[C#] ITableMapping ITableMappingCollection.Add(string sourceTableName,
string
dataSetTableName);
[C++] ITableMapping* ITableMappingCollection::Add(String*
sourceTableName, String*
dataSetTableName);
[VB] Function Add(ByVal sourceTableName As String, ByVal
dataSetTableName As String) As ITableMapping Implements
ITableMappingCollection.Add
```

```

1 [JScript] function ITableMappingCollection.Add(sourceTableName : String,
2 dataSetTableName : String) : ITableMapping;
3     ITableMappingCollection.GetByDataSetTable
4
5 [C#] ITableMapping ITableMappingCollection.GetByDataSetTable(string
6 dataSetTableName);
7 [C++] ITableMapping* ITableMappingCollection::GetByDataSetTable(String*
8 dataSetTableName);
9 [VB] Function GetByDataSetTable(ByVal dataSetTableName As String) As
10 ITableMapping Implements ITableMappingCollection.GetByDataSetTable
11 [JScript] function
12 ITableMappingCollection.GetByDataSetTable(dataSetTableName : String) :
13 ITableMapping;
14     DbDataAdapter class (System.Data.Common)
15     ToString
16
17

```

## 18 *Description*

19 Aids implementation of the **System.Data.IDbDataAdapter** interface.

20 Inheritors of **System.Data.Common.DbDataAdapter** implement a set of

21 functions to provide strong typing, but inherit most of the functionality needed to

22 fully implement a DataAdapter.

23 The **System.Data.Common.DbDataAdapter** class inherits from the

24 **System.Data.Common.DataAdapter** class and helps a class implement a

25 DataAdapter designed for use with a relational database.

```

1      ToString
2
3  [C#]      public      const      string      DefaultSourceTableName;
4  [C++]     public:     const      String*      DefaultSourceTableName;
5  [VB]      Public     Const      DefaultSourceTableName      As      String
6  [JScript] public     var      DefaultSourceTableName      :      String;
7

```

### 8 *Description*

9       The default name used by the **System.Data.Common.DataAdapter** object  
10      for table mappings.

11       **System.Data.Common.DbDataAdapter.DefaultSourceTableName** is  
12      when an application adds a table mapping to be used with  
13      **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** , but  
14      does not specify a **System.Data.DataTable** name.

15      DbDataAdapter

### 16 *Example Syntax:*

17      ToString

```

18
19  [C#]      protected      DbDataAdapter();
20  [C++]     protected:     DbDataAdapter();
21  [VB]      Protected      Sub      New()
22  [JScript] protected      function DbDataAdapter();
23

```

### 24 *Description*

25

1        Initializes a new instance of the **System.Data.Common.DbDataAdapter**  
2 class.

3        When you create an instance of **System.Data.Common.DbDataAdapter** ,  
4 the following read/write properties are set to the following initial values.

- 5        **AcceptChangesDuringFill**
- 6        **Container**
- 7        **DesignMode**
- 8        **Events**
- 9        **MissingMappingAction**
- 10       **MissingSchemaAction**
- 11       **Site**
- 12       **TableMappings**
- 13       **ToString**

16       *Description*

17        Returned when an error occurs during a fill operation.

18        The **System.Data.Common.DbDataAdapter.FillError** event allows a  
19 user to determine whether or not the fill operation should continue after the error  
20 occurs. Examples of when the **System.Data.Common.DbDataAdapter.FillError**  
21 event might occur are: The data being added to a **System.Data.DataSet** cannot be  
22 converted to a common language runtime type without losing precision.

23        **CreateRowUpdatedEvent**

25        [C#]                      protected                      abstract                      RowUpdatedEventArgs

```

1 CreateRowUpdatedEvent(DataRow    dataRow,    IDbCommand    command,
2 StatementType    statementType,    DataTableMapping    tableMapping);
3 [C++]    protected:    virtual    RowUpdatedEventArgs*
4 CreateRowUpdatedEvent(DataRow*    dataRow,    IDbCommand*    command,
5 StatementType    statementType,    DataTableMapping*    tableMapping) = 0;
6 [VB] MustOverride Protected Function CreateRowUpdatedEvent(ByVal dataRow
7 As DataRow, ByVal command As IDbCommand, ByVal statementType As
8 StatementType; ByVal tableMapping As DataTableMapping) As
9 RowUpdatedEventArgs
10 [JScript] protected abstract function CreateRowUpdatedEvent(dataRow :
11 DataRow, command : IDbCommand, statementType : StatementType,
12 tableMapping : DataTableMapping) : RowUpdatedEventArgs;

```

#### 14 *Description*

15 Initializes a new instance of the  
16 **System.Data.Common.RowUpdatedEventArgs** class.

17 *Return Value:* A new instance of the  
18 **System.Data.Common.RowUpdatedEventArgs** class.

19 When overriding  
20 **System.Data.Common.DbDataAdapter.CreateRowUpdatedEvent(System.Data  
21 a.DataRow,System.Data.IDbCommand,System.Data.StatementType,System.  
22 Data.Common.DataTableMapping)** in a derived class, be sure to call the base  
23 class's

24 **System.Data.Common.DbDataAdapter.CreateRowUpdatedEvent(System.Data  
25 a.DataRow,System.Data.IDbCommand,System.Data.StatementType,System.**

**Data.Common.DataTableMapping)** method. The **System.Data.DataRow** used to update the data source. The **System.Data.IDbCommand** executed during the **System.Data.IDataAdapter.Update(System.Data.DataSet)**. Whether the command is an UPDATE, INSERT, DELETE, or SELECT statement. A **System.Data.Common.DataTableMapping** object.

#### CreateRowUpdatingEvent

```
[C#]           protected           abstract           RowUpdatingEventArgs
CreateRowUpdatingEvent(DataRow  dataRow,  IDbCommand  command,
StatementType  statementType,  DataTableMapping  tableMapping);

[C++]           protected:           virtual           RowUpdatingEventArgs*
CreateRowUpdatingEvent(DataRow*  dataRow,  IDbCommand*  command,
StatementType  statementType,  DataTableMapping*  tableMapping) = 0;

[VB]  MustOverride  Protected  Function  CreateRowUpdatingEvent(ByVal
dataRow As DataRow, ByVal command As IDbCommand, ByVal statementType
As StatementType, ByVal tableMapping As DataTableMapping) As
RowUpdatingEventArgs

[JScript] protected abstract function CreateRowUpdatingEvent(dataRow :
DataRow, command : IDbCommand, statementType : StatementType,
tableMapping : DataTableMapping) : RowUpdatingEventArgs;
```

#### *Description*

Initializes a new instance of the **System.Data.Common.RowUpdatingEventArgs** class.

*Return Value:* A new instance of the **System.Data.Common.RowUpdatingEventArgs** class.

When overriding **System.Data.Common.DbDataAdapter.CreateRowUpdatingEvent(System.Data.DataRow, System.Data.IDbCommand, System.Data.StatementType, System.Data.Common.DataTableMapping)** in a derived class, be sure to call the base class's

**System.Data.Common.DbDataAdapter.CreateRowUpdatingEvent(System.Data.DataRow, System.Data.IDbCommand, System.Data.StatementType, System.Data.Common.DataTableMapping)** method. The **System.Data.DataRow** that updates the data source. The **System.Data.IDbCommand** to execute during the **System.Data.IDataAdapter.Update(System.Data.DataSet)**. Whether the command is an UPDATE, INSERT, DELETE, or SELECT statement. A **System.Data.Common.DataTableMapping** object.

Fill

[C#] public override int Fill(DataSet dataSet);

[C++] public: int Fill(DataSet\* dataSet);

[VB] Overrides Public Function Fill(ByVal dataSet As DataSet) As Integer

[JScript] public override function Fill(dataSet : DataSet) : int;

### *Description*

Adds or refreshes rows in the **System.Data.DataSet** to match those in the data source using the **System.Data.DataSet** name, and creates a **System.Data.DataTable** named "Table".

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataSet** . This does not include rows affected by statements that do not return rows.

The **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method retrieves the data from the data source using a SELECT statement. The **System.Data.IDbConnection** object associated with the select command must be valid, but it does not need to be open. If the **System.Data.IDbConnection** is closed before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it remains open. A **System.Data.DataSet** to fill with records and, if necessary, schema.

#### Fill

[C#]            public            int            Fill(DataTable            dataTable);  
[C++]           public:           int           Fill(DataTable\*           dataTable);  
[VB]   Public   Function   Fill(ByVal   dataTable   As   DataTable)   As   Integer  
[JScript]   public   function   Fill(dataTable : DataTable) : int; Adds or refreshes rows in the **System.Data.DataSet** to match those in the data source.

#### *Description*

Adds or refreshes rows in a **System.Data.DataTable** to match those in the data source using the **System.Data.DataTable** name.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataTable** . This does not include rows affected by statements that do not return rows.

The **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method retrieves rows from the data source using the SELECT statement specified by an associated **System.Data.IDbDataAdapter.SelectCommand** property. The connection object associated with the SELECT statement must be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it remains open. A **System.Data.DataTable** to fill with records and, if necessary, schema.

#### Fill

[C#] public int Fill(DataSet dataSet, string srcTable);

[C++] public: int Fill(DataSet\* dataSet, String\* srcTable);

[VB] Public Function Fill(ByVal dataSet As DataSet, ByVal srcTable As String)

As Integer

[JScript] public function Fill(dataSet : DataSet, srcTable : String) : int;

#### Description

Adds or refreshes rows in the **System.Data.DataSet** to match those in the data source using the **System.Data.DataSet** and **System.Data.DataTable** names.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataSet** . This does not include rows affected by statements that do not return rows.

The **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method retrieves the data from the data source using a SELECT statement. The **System.Data.IDbConnection** object associated with the select command must be valid, but it does not need to be open. If the **System.Data.IDbConnection** is closed before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it remains open. A **System.Data.DataSet** to fill with records and, if necessary, schema. The name of the source table to use for table mapping.

#### Fill

[C#] protected virtual int Fill(DataTable dataTable, IDataReader dataReader);

[C++] protected: virtual int Fill(DataTable\* dataTable, IDataReader\* dataReader);

[VB] Overridable Protected Function Fill(ByVal dataTable As DataTable, ByVal dataReader As IDataReader) As Integer

[JScript] protected function Fill(dataTable : DataTable, dataReader : IDataReader)

: int;

#### Description

Adds or refreshes rows in a **System.Data.DataTable** to match those in the data source using the specified **System.Data.DataTable** and **System.Data.IDataReader** names.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataTable** . This does not include rows affected by statements that do not return rows.

The **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method retrieves rows from the data source using the SELECT statement specified by an associated **System.Data.IDbDataAdapter.SelectCommand** property. The connection object associated with the SELECT statement must be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it remains open. A **System.Data.DataTable** to fill with records and, if necessary, schema. The name of the **System.Data.IDataReader** .

Fill

[C#] protected virtual int Fill(DataTable dataTable, IDbCommand command, CommandBehavior behavior);

[C++] protected: virtual int Fill(DataTable\* dataTable, IDbCommand\* command, CommandBehavior behavior);

[VB] Overridable Protected Function Fill(ByVal dataTable As DataTable, ByVal command As IDbCommand, ByVal behavior As CommandBehavior) As Integer

[JScript] protected function Fill(dataTable : DataTable, command : IDbCommand,  
behavior : CommandBehavior) : int;

#### *Description*

Adds or refreshes rows in a **System.Data.DataTable** to match those in the data source using the **System.Data.DataTable** name, the specified SQL SELECT statement, and **System.Data.CommandBehavior**.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataTable**. This does not include rows affected by statements that do not return rows.

The **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method retrieves rows from the data source using the SELECT statement specified by an associated **System.Data.IDbDataAdapter.SelectCommand** property. The connection object associated with the SELECT statement must be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it remains open. A **System.Data.DataTable** to fill with records and, if necessary, schema. The SQL SELECT statement used to retrieve rows from the data source. One of the the **System.Data.CommandBehavior** values.

Fill

[C#] public int Fill(DataSet dataSet, int startRecord, int maxRecords, string

```

1  srcTable);
2  [C++] public: int Fill(DataSet* dataSet, int startRecord, int maxRecords, String*
3  srcTable);
4  [VB] Public Function Fill(ByVal dataSet As DataSet, ByVal startRecord As
5  Integer, ByVal maxRecords As Integer, ByVal srcTable As String) As Integer
6  [JScript] public function Fill(dataSet : DataSet, startRecord : int, maxRecords : int,
7  srcTable          :          String)          :          int;
8

```

### *Description*

Adds or refreshes rows in a specified range in the **System.Data.DataSet** to match those in the data source using the **System.Data.DataSet** and **System.Data.DataTable** names.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataSet**. This does not include rows affected by statements that do not return rows.

A *maxRecords* value of 0 gets all records found after the start record. If *maxRecords* is greater than the number of remaining rows, only the remaining rows are returned and no error is issued. A **System.Data.DataSet** to fill with records and, if necessary, schema. The zero-based record number to start with. The maximum number of records to retrieve. The name of the source table to use for table mapping.

### Fill

```

22
23
24  [C#] protected virtual int Fill(DataSet dataSet, string srcTable, IDataReader
25  dataReader,          int          startRecord,          int          maxRecords);

```

```

1  [C++] protected: virtual int Fill(DataSet* dataSet, String* srcTable, IDataReader*
2  dataReader,          int          startRecord,          int          maxRecords);
3  [VB] Overridable Protected Function Fill(ByVal dataSet As DataSet, ByVal
4  srcTable As String, ByVal dataReader As IDataReader, ByVal startRecord As
5  Integer,          ByVal          maxRecords          As          Integer)          As          Integer
6  [JScript] protected function Fill(dataSet : DataSet, srcTable : String, dataReader :
7  IDataReader, startRecord : int, maxRecords : int) : int;
8

```

### *Description*

Adds or refreshes rows in a specified range in the **System.Data.DataSet** to match those in the data source using the **System.Data.DataSet** , **System.Data.DataTable** , and **System.Data.IDataReader** names.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataSet** . This does not include rows affected by statements that do not return rows.

The **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method retrieves rows from the data source using the SELECT statement specified by an associated **System.Data.IDbDataAdapter.SelectCommand** property. The connection object associated with the SELECT statement must be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** is called, it remains open. A **System.Data.DataSet** to fill with records and, if necessary,

schema. The name of the **System.Data.DataTable** to use for table mapping. The name of the **System.Data.IDataReader**. The zero-based record number to start with. The maximum number of records to retrieve.

#### Fill

[C#] protected virtual int Fill(DataSet dataSet, int startRecord, int maxRecords, string srcTable, IDbCommand command, CommandBehavior behavior);

[C++] protected: virtual int Fill(DataSet\* dataSet, int startRecord, int maxRecords, String\* srcTable, IDbCommand\* command, CommandBehavior behavior);

[VB] Overridable Protected Function Fill(ByVal dataSet As DataSet, ByVal startRecord As Integer, ByVal maxRecords As Integer, ByVal srcTable As String, ByVal command As IDbCommand, ByVal behavior As CommandBehavior) As Integer

[JScript] protected function Fill(dataSet : DataSet, startRecord : int, maxRecords : int, srcTable : String, command : IDbCommand, behavior : CommandBehavior) : int;

#### *Description*

Adds or refreshes rows in a specified range in the **System.Data.DataSet** to match those in the data source using the **System.Data.DataSet** and source table names, command string and command behavior.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataSet** . This does not include rows affected by statements that do not return rows.

1000 900 800 700 600 500 400 300 200 100 0

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

17

18

19

20

21

22

23

24

1  
2 *Description*

3 Adds a **System.Data.DataTable** named "Table" to the specified  
4 **System.Data.DataSet** and configures the schema to match that in the data source  
5 based on the specified **System.Data.SchemaType**.

6 *Return Value:* A reference to a collection of **System.Data.DataTable** objects that  
7 were added to the **System.Data.DataSet**.

8 This method retrieves the schema information from the data source using  
9 the **System.Data.IDbDataAdapter.SelectCommand**. A **System.Data.DataSet**  
10 to insert the schema in. One of the **System.Data.SchemaType** values that specify  
11 how to insert the schema.

12 **FillSchema**

13  
14 [C#] public DataTable FillSchema(DataTable dataTable, SchemaType  
15 schemaType);

16 [C++] public: DataTable\* FillSchema(DataTable\* dataTable, SchemaType  
17 schemaType);

18 [VB] Public Function FillSchema(ByVal dataTable As DataTable, ByVal  
19 schemaType As SchemaType) As DataTable

20 [JScript] public function FillSchema(dataTable : DataTable, schemaType :  
21 SchemaType) : DataTable; Adds a **System.Data.DataTable** to a  
22 **System.Data.DataSet** and configures the schema to match that in the data source.

23  
24 *Description*  
25

Configures the schema of the specified **System.Data.DataTable** based on the specified **System.Data.SchemaType** .

*Return Value:* A **System.Data.DataTable** that contains schema information returned from the data source.

The **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable, System.Data.SchemaType)** method retrieves the schema from the data source using the **System.Data.IDbDataAdapter.SelectCommand** . The connection object associated with the **System.Data.IDbDataAdapter.SelectCommand** must be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable, System.Data.SchemaType)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable, System.Data.SchemaType)** is called, it remains open. The **System.Data.DataTable** to be filled with the schema from the data source. One of the **System.Data.SchemaType** values.

#### FillSchema

```
[C#] public DataTable[] FillSchema(DataSet dataSet, SchemaType schemaType,
string srcTable);
```

```
[C++] public: DataTable* FillSchema(DataSet* dataSet, SchemaType
schemaType, String* srcTable) [];
```

```
[VB] Public Function FillSchema(ByVal dataSet As DataSet, ByVal schemaType
As SchemaType, ByVal srcTable As String) As DataTable()
```

```
[JScript] public function FillSchema(dataSet : DataSet, schemaType :
SchemaType, srcTable : String) : DataTable[];
```

#### *Description*

Adds a **System.Data.DataTable** to the specified **System.Data.DataSet** and configures the schema to match that in the data source based upon the specified **System.Data.SchemaType** and **System.Data.DataTable**.

*Return Value:* A reference to a collection of **System.Data.DataTable** objects that were added to the **System.Data.DataSet**.

This method retrieves the schema information from the data source using the **System.Data.IDbDataAdapter.SelectCommand**. A **System.Data.DataSet** to insert the schema in. One of the **System.Data.SchemaType** values that specify how to insert the schema. The name of the source table to use for table mapping.

#### FillSchema

```
[C#] protected virtual DataTable FillSchema(DataTable dataTable, SchemaType
schemaType, IDbCommand command, CommandBehavior behavior);
```

```
[C++] protected: virtual DataTable* FillSchema(DataTable* dataTable,
SchemaType schemaType, IDbCommand* command, CommandBehavior
behavior);
```

```
[VB] Overridable Protected Function FillSchema(ByVal dataTable As DataTable,
ByVal schemaType As SchemaType, ByVal command As IDbCommand, ByVal
behavior As CommandBehavior) As DataTable
```

```
[JScript] protected function FillSchema(dataTable : DataTable, schemaType :
SchemaType, command : IDbCommand, behavior : CommandBehavior) :
```

1 DataTable;

2  
3 *Description*

4 Adds a **System.Data.DataTable** to a **System.Data.DataSet** and configures  
5 the schema to match that in the data source based on the specified

6 **System.Data.SchemaType**

7 *Return Value:* An array of **System.Data.DataTable** objects that contain schema  
8 information returned from the data source.

9 The  
10 **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable,S**  
11 **ystem.Data.SchemaType)** method retrieves the schema from the data source  
12 using the **System.Data.IDbDataAdapter.SelectCommand** . The connection  
13 object associated with the **System.Data.IDbDataAdapter.SelectCommand** must  
14 be valid, but it does not need to be open. If the connection is closed before  
15 **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable,S**  
16 **ystem.Data.SchemaType)** is called, it is opened to retrieve data, then closed. If  
17 the connection is open before  
18 **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable,S**  
19 **ystem.Data.SchemaType)** is called, it remains open. The  
20 **System.Data.DataTable** to be filled with the schema from the data source. One of  
21 the **System.Data.SchemaType** values. The SQL SELECT statement used to  
22 retrieve rows from the data source. One of the the  
23 **System.Data.CommandBehavior** values.

24 FillSchema  
25

```

1
2 [C#] protected virtual DataTable[] FillSchema(DataSet dataSet, SchemaType
3 schemaType, IDbCommand command, string srcTable, CommandBehavior
4 behavior);
5 [C++] protected: virtual DataTable* FillSchema(DataSet* dataSet, SchemaType
6 schemaType, IDbCommand* command, String* srcTable, CommandBehavior
7 behavior)                                     [];
8 [VB] Overridable Protected Function FillSchema(ByVal dataSet As DataSet,
9 ByVal schemaType As SchemaType, ByVal command As IDbCommand, ByVal
10 srcTable As String, ByVal behavior As CommandBehavior) As DataTable()
11 [JScript] protected function FillSchema(dataSet : DataSet, schemaType :
12 SchemaType, command : IDbCommand, srcTable : String, behavior :
13 CommandBehavior)                               :           DataTable[];
14

```

#### *Description*

Adds a **System.Data.DataTable** to the specified **System.Data.DataSet** and configures the schema to match that in the data source based on the specified **System.Data.SchemaType**.

*Return Value:* An array of **System.Data.DataTable** objects that contain schema information returned from the data source.

The **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable, System.Data.SchemaType)** method retrieves the schema from the data source using the **System.Data.IDbDataAdapter.SelectCommand**. The connection object associated with the **System.Data.IDbDataAdapter.SelectCommand** must

be valid, but it does not need to be open. If the connection is closed before **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable, System.Data.SchemaType)** is called, it is opened to retrieve data, then closed. If the connection is open before **System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataTable, System.Data.SchemaType)** is called, it remains open. The **System.Data.DataSet** to be filled with the schema from the data source. One of the **System.Data.SchemaType** values. The SQL SELECT statement used to retrieve rows from the data source. The name of the source table to use for table mapping. One of the **System.Data.CommandBehavior** values.

#### GetFillParameters

```
[C#]      public      override      IDataParameter[]      GetFillParameters();
[C++]      public:      IDataParameter*      GetFillParameters()      [];
[VB]      Overrides      Public      Function      GetFillParameters()      As      IDataParameter()
[JScript]      public      override      function      GetFillParameters()      :      IDataParameter[];
```

#### *Description*

Gets the parameters set by the user when executing an SQL SELECT statement.

*Return Value:* An array of **System.Data.IDataParameter** objects that contains the parameters set by the user.

#### OnFillError

```
[C#]      protected      virtual      void      OnFillError(FillErrorEventArgs      value);
```

1 [C++] protected: virtual void OnFillError(FillEventArgs\* value);  
2 [VB] Overridable Protected Sub OnFillError(ByVal value As FillEventArgs)  
3 [JScript] protected function OnFillError(value : FillEventArgs);  
4

5 *Description*

6 Raises the **System.Data.Common.DbDataAdapter.FillError** event.

7 Raising an event invokes the event handler through a delegate. For an  
8 overview, see . A **System.Data.FillEventArgs** that contains the event data.

9 **OnRowUpdated**

10  
11 [C#] protected abstract void OnRowUpdated(RowUpdatedEventArgs value);

12 [C++] protected: virtual void OnRowUpdated(RowUpdatedEventArgs\* value) =  
13 0;

14 [VB] MustOverride Protected Sub OnRowUpdated(ByVal value As  
15 RowUpdatedEventArgs)

16 [JScript] protected abstract function OnRowUpdated(value :  
17 RowUpdatedEventArgs);  
18

19 *Description*

20 Raises the **RowUpdated** event of a .NET data provider.

21 Raising an event invokes the event handler through a delegate. For an  
22 overview, see . A **System.Data.Common.RowUpdatedEventArgs** that contains  
23 the event data.

24 **OnRowUpdating**  
25

```

1
2 [C#] protected abstract void OnRowUpdating(RowUpdatingEventArgs value);
3 [C++] protected: virtual void OnRowUpdating(RowUpdatingEventArgs* value) =
4 0;
5 [VB] MustOverride Protected Sub OnRowUpdating(ByVal value As
6 RowUpdatingEventArgs)
7 [JScript] protected abstract function OnRowUpdating(value :
8 RowUpdatingEventArgs);
9

```

#### 10 *Description*

11 Raises the **RowUpdating** event of a .NET data provider.

12 Raising an event invokes the event handler through a delegate. For an  
13 overview, see . An **System.Data.OleDb.OleDbRowUpdatingEventArgs** that  
14 contains the event data.

#### 15 *Update*

```

16
17 [C#] public int Update(DataRow[] dataRows);
18 [C++] public: int Update(DataRow* dataRows[]);
19 [VB] Public Function Update(ByVal dataRows() As DataRow) As Integer
20 [JScript] public function Update(dataRows : DataRow[]) : int;
21

```

#### 22 *Description*

23 Calls the respective INSERT, UPDATE, or DELETE statements for each  
24 inserted, updated, or deleted row in the specified array of **System.Data.DataRow**  
25 objects.

*Return Value:* The number of rows successfully updated from the **System.Data.DataSet**.

When an application calls the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** method, the **System.Data.Common.DbDataAdapter** examines the **System.Data.DataRow.RowState** property, and executes the required INSERT, UPDATE, or DELETE statements based on the order of the indexes configured in the **System.Data.DataSet**. For example, **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** might execute a DELETE statement, followed by an INSERT statement, and then another DELETE statement, due to the ordering of the rows in the **System.Data.DataTable**. An application can call the **System.Data.DataSet.GetChanges** method in situations where you must control the sequence of statement types (for example, INSERTs before UPDATEs). For more information, see . An array of **System.Data.DataRow** objects used to update the data source.

#### Update

```
[C#]      public override int Update(DataSet dataSet);
[C++]      public: int Update(DataSet* dataSet);
[VB] Overrides Public Function Update(ByVal dataSet As DataSet) As Integer
[JScript] public override function Update(dataSet : DataSet) : int; Calls the
respective INSERT, UPDATE, or DELETE statements for each inserted, updated,
or deleted row in the System.Data.DataSet from a System.Data.DataTable
named "Table".
```

## Description

Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the specified **System.Data.DataSet**.

**Return Value:** The number of rows successfully updated from the **System.Data.DataSet**.

When an application calls the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** method, the **System.Data.Common.DbDataAdapter** examines the **System.Data.DataRow.RowState** property, and executes the required INSERT, UPDATE, or DELETE statements based on the order of the indexes configured in the **System.Data.DataSet**. For example, **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** might execute a DELETE statement, followed by an INSERT statement, and then another DELETE statement, due to the ordering of the rows in the **System.Data.DataTable**. An application can call the **System.Data.DataSet.GetChanges** method in situations where you must control the sequence of statement types (for example, INSERTs before UPDATEs). For more information, see . The **System.Data.DataSet** used to update the data source.

## Update

```
[C#]      public      int      Update(DataTable      dataTable);  
[C++]     public:     int      Update(DataTable*      dataTable);  
[VB]      Public Function Update(ByVal dataTable As DataTable) As Integer  
[JScript] public      function Update(dataTable : DataTable) : int;
```

## Description

Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the specified **System.Data.DataTable** .

*Return Value:* The number of rows successfully updated from the **System.Data.DataSet** .

When an application calls the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** method, the **System.Data.Common.DbDataAdapter** examines the **System.Data.DataRow.RowState** property, and executes the required INSERT, UPDATE, or DELETE statements based on the order of the indexes configured in the **System.Data.DataSet** . For example, **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** might execute a DELETE statement, followed by an INSERT statement, and then another DELETE statement, due to the ordering of the rows in the **System.Data.DataTable** . An application can call the **System.Data.DataSet.GetChanges** method in situations where you must control the sequence of statement types (for example, INSERTs before UPDATEs). For more information, see . The **System.Data.DataTable** used to update the data source.

## Update

[C#] protected virtual int Update(DataRow[] dataRows, DataTableMapping tableMapping);

[C++] protected: virtual int Update(DataRow\* dataRows[], DataTableMapping\*

```

1 tableMapping);
2 [VB] Overridable Protected Function Update(ByVal dataRows() As DataRow,
3 ByVal tableMapping As DataTableMapping) As Integer
4 [JScript] protected function Update(dataRows : DataRow[], tableMapping :
5 DataTableMapping) : int;
6

```

### *Description*

Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the specified array of **System.Data.DataRow** objects.

*Return Value:* The number of rows successfully updated from the **System.Data.DataSet**.

When an application calls the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** method, the **System.Data.Common.DbDataAdapter** examines the **System.Data.DataRow.RowState** property, and executes the required INSERT, UPDATE, or DELETE statements based on the order of the indexes configured in the **System.Data.DataSet**. For example, **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** might execute a DELETE statement, followed by an INSERT statement, and then another DELETE statement, due to the ordering of the rows in the **System.Data.DataTable**. An application can call the **System.Data.DataSet.GetChanges** method in situations where you must control the sequence of statement types (for example, INSERTs before UPDATEs). For more information, see . An array of **System.Data.DataRow** objects used to update

the data source. The **System.Data.IDataAdapter.TableMappings** collection to use.

### Update

```
[C#]    public    int    Update(DataSet    dataSet,    string    srcTable);
[C++]   public:    int    Update(DataSet*    dataSet,    String*    srcTable);
[VB]    Public Function Update(ByVal dataSet As DataSet, ByVal srcTable As
String)                                     As                                     Integer
[JScript] public function Update(dataSet : DataSet, srcTable : String) : int;
```

### *Description*

Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the **System.Data.DataSet** with the specified **System.Data.DataTable** name.

*Return Value:* The number of rows successfully updated from the **System.Data.DataSet**.

When an application calls the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** method, the **System.Data.Common.DbDataAdapter** examines the **System.Data.DataRow.RowState** property, and executes the required INSERT, UPDATE, or DELETE statements based on the order of the indexes configured in the **System.Data.DataSet**. For example, **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** might execute a DELETE statement, followed by an INSERT statement, and then another DELETE statement, due to the ordering of the rows in the

1 **System.Data.DataTable** . An application can call the  
 2 **System.Data.DataSet.GetChanges** method in situations where you must control  
 3 the sequence of statement types (for example, INSERTs before UPDATEs). For  
 4 more information, see . The **System.Data.DataSet** to use to update the data  
 5 source. The name of the source table to use for table mapping.

6 DBDataPermission class (System.Data.Common)

7 Update

8  
 9  
 10 *Description*

11 Provides the capability for a .NET data provider to ensure that a user has a  
 12 security level adequate for accessing data.

13 DBDataPermission

14 *Example Syntax:*

15 Update

16  
 17 [C#] protected DBDataPermission();

18 [C++] protected: DBDataPermission();

19 [VB] Protected Sub New()

20 [JScript] protected function DBDataPermission(); Initializes a new instance of the

21 **System.Data.Common.DBDataPermission** class.

22  
 23 *Description*

24 Initializes a new instance of the

25 **System.Data.Common.DBDataPermission** class.

DBDataPermission

*Example Syntax:*

Update

[C#]       protected       DBDataPermission(PermissionState       state);

[C++]       protected:       DBDataPermission(PermissionState       state);

[VB]   Protected   Sub   New(ByVal   state   As   PermissionState)

[JScript]   protected   function   DBDataPermission(state   :   PermissionState);

*Description*

Initializes       a       new       instance       of       the  
**System.Data.Common.DBDataPermission**       class.       One       of       the  
**System.Security.Permissions.PermissionState** values.

DBDataPermission

*Example Syntax:*

Update

[C#] public DBDataPermission(PermissionState state, bool allowBlankPassword);

[C++]       public:       DBDataPermission(PermissionState       state,       bool  
allowBlankPassword);

[VB]   Public   Sub   New(ByVal   state   As   PermissionState,   ByVal  
allowBlankPassword                               As               Boolean)

[JScript]   public   function   DBDataPermission(state   :   PermissionState,  
allowBlankPassword                               :               Boolean);

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Initializes a new instance of the **System.Data.Common.DBDataPermission** class. Indicates whether a blank password is allowed.

AllowBlankPassword

Update

```
[C#]      public      bool      AllowBlankPassword      {get;      set;}
[C++] public: __property bool get_AllowBlankPassword();public: __property
void set_AllowBlankPassword(bool);
[VB]      Public      Property      AllowBlankPassword      As      Boolean
[JScript] public function get AllowBlankPassword() : Boolean;public function set
AllowBlankPassword(Boolean);
```

*Description*

Gets a value indicating whether a blank password is allowed.

Copy

```
[C#]      public      override      IPermission      Copy();
[C++]      public:      IPermission*      Copy();
[VB]      Overrides      Public      Function      Copy()      As      IPermission
[JScript] public      override      function      Copy()      :      IPermission;
```

*Description*

Creates and returns an identical copy of the current permission object.

*Return Value:* A copy of the current permission object.

A copy of a permission object represents the same access to resources as the original permission object.

### FromXml

[C#] public override void FromXml(SecurityElement securityElement);

[C++] public: void FromXml(SecurityElement\* securityElement);

[VB] Overrides Public Sub FromXml(ByVal securityElement As SecurityElement)

[JScript] public override function FromXml(securityElement : SecurityElement);

### Description

Reconstructs a security object with a specified state from an XML encoding.

Custom code that extends security objects needs to implement the ToXml and **FromXml** methods to make the objects security-encodable. The XML encoding to use to reconstruct the security object.

### Intersect

[C#] public override IPermission Intersect(IPermission target);

[C++] public: IPermission\* Intersect(IPermission\* target);

[VB] Overrides Public Function Intersect(ByVal target As IPermission) As IPermission

[JScript] public override function Intersect(target : IPermission) : IPermission;

## Description

Returns a new permission object representing the intersection of the current permission object and the specified permission object.

**Return Value:** A new permission object that represents the intersection of the current permission object and the specified permission object. This new permission object is a null reference ( **Nothing** in Visual Basic) if the intersection is empty. The *target* parameter is not a null reference ( **Nothing** in Visual Basic) and is not an instance of the same class as the current permission object.

The intersection of two permissions is a permission that describes the set of operations they both describe in common. Only a demand that passes both original permissions will pass the intersection. A permission object to intersect with the current permission object. It must be of the same type as the current permission object.

## IsSubsetOf

[C#] public override bool IsSubsetOf(IPermission target);

[C++] public: bool IsSubsetOf(IPermission\* target);

[VB] Overrides Public Function IsSubsetOf(ByVal target As IPermission) As Boolean

[JScript] public override function IsSubsetOf(target : IPermission) : Boolean;

## Description

Returns a value indicating whether the current permission object is a subset of the specified permission object.

*Return Value:* **True** if the current permission object is a subset of the specified permission object; otherwise **false** .

The current permission object is a subset of the specified permission object if the current permission object specifies a set of operations that is wholly contained by the specified permission object. For example, a permission that represents access to C:\example.txt is a subset of a permission that represents access to C:\. If this method returns **true** , the current permission object represents no more access to the protected resource than does the specified permission object. A permission object that is to be tested for the subset relationship. This object must be of the same type as the current permission object.

IsUnrestricted

[C#]	public	bool	IsUnrestricted();
[C++]	public:	__sealed	bool IsUnrestricted();
[VB]	NotOverridable	Public Function	IsUnrestricted() As Boolean
[JScript]	public	function	IsUnrestricted() : Boolean;

### *Description*

Returns a value indicating whether the permission can be represented as unrestricted without any knowledge of the permission semantics.

*Return Value:* **True** if the permission can be represented as unrestricted.

This is a binary permission; therefore the implementation always returns **true** .

ToXml

```

1
2 [C#]      public      override      SecurityElement      ToXml();
3 [C++]      public:      SecurityElement*      ToXml();
4 [VB]  Overrides  Public  Function  ToXml()  As  SecurityElement
5 [JScript]  public  override  function  ToXml()  :  SecurityElement;

```

### Description

Creates an XML encoding of the security object and its current state.

*Return Value:* An XML encoding of the security object, including any state information.

Custom code that extends security objects needs to implement the **System.Data.Common.DBDataPermission.ToXml** and **System.Data.Common.DBDataPermission.FromXml(System.Security.SecurityElement)** methods to make the objects security-encodable.

### Union

```

17 [C#]      public      override      IPermission      Union(IPermission      target);
18 [C++]      public:      IPermission*      Union(IPermission*      target);
19 [VB]  Overrides  Public  Function  Union(ByVal target As IPermission) As
20 IPermission
21 [JScript]  public  override  function  Union(target : IPermission) : IPermission;

```

### Description

Returns a new permission object that is the union of the current and specified permission objects.

1 *Return Value:* A new permission object that represents the union of the current  
2 permission object and the specified permission object.

3 The result of a call to  
4 **System.Data.Common.DBDataPermission.Union(System.Security.IPermissio**  
5 **n)** is a permission that represents all of the operations represented by both the  
6 current permission object and the specified permission object. Any demand that  
7 passes either permission passes their union. A permission object to combine with  
8 the current permission object. It must be of the same type as the current  
9 permission object.

10 DBDataPermissionAttribute class (System.Data.Common)

11 Union

12  
13  
14 *Description*

15 Associates a security action with a custom security attribute.

16 DBDataPermissionAttribute

17 *Example Syntax:*

18 Union

19  
20 [C#] protected DBDataPermissionAttribute(SecurityAction action);

21 [C++] protected: DBDataPermissionAttribute(SecurityAction action);

22 [VB] Protected Sub New(ByVal action As SecurityAction)

23 [JScript] protected function DBDataPermissionAttribute(action : SecurityAction);

24  
25 *Description*

1        Initializes        a        new        instance        of        the  
2        **System.Data.Common.DBDataPermissionAttribute**        class.

3        *Return Value:* A **System.Data.Common.DBDataPermissionAttribute** object.

4        One of the the **System.Security.Permissions.SecurityAction** values representing  
5        an action that can be performed using declarative security.

6        Action

7        AllowBlankPassword

8        Union

11        *Description*

12        Gets a value indicating whether a blank password is allowed.

13        TypeId

14        Unrestricted

15        DbDataRecord class (System.Data.Common)

16        ToString

19        *Description*

20        FieldCount

21        ToString

23        [C#]        public        int        FieldCount        {get;}

24        [C++]        public:        \_\_property        int        get\_FieldCount();

25        [VB]        Public        ReadOnly        Property        FieldCount        As Integer

1 [JScript]      public      function      get      FieldCount()      :      int;

2

3 *Description*

4            Indicates the number of fields within the current record. This property is  
5 read-only.

6            Item

7            ToString

8

9 [C#]            public            object            this[string            name]            {get;}

10 [C++]          public:          \_\_property          Object\*          get\_Item(String\*          name);

11 [VB] Public Default ReadOnly Property Item(ByVal name As String) As Object

12 [JScript]          returnValue          =          DbDataRecordObject.Item(name);

13

14 *Description*

15            Indicates the value at the specified column in its native format given the  
16 column name. This property is read-only. The column name.

17            Item

18            ToString

19

20 [C#]            public            object            this[int            i]            {get;}

21 [C++]          public:          \_\_property          Object\*          get\_Item(int          i);

22 [VB] Public Default ReadOnly Property Item(ByVal i As Integer) As Object

23 [JScript] returnValue = DbDataRecordObject.Item(i); Indicates that value from a  
24 column in its native format. This property is read-only.

25

## Description

Indicates the value at the specified column in its native format given the column ordinal. This property is read-only. The column ordinal.

### GetBoolean

```
[C#]          public          bool          GetBoolean(int          i);
[C++]        public:    __sealed    bool          GetBoolean(int    i);
[VB] NotOverridable Public Function GetBoolean(ByVal i As Integer) As
Boolean
[JScript]    public    function    GetBoolean(i    :    int)    :    Boolean;
```

## Description

Returns the value of the specified column as a boolean.

**Return Value:** **true** if the boolean is **true** ; otherwise, **false** .

No conversions are performed, therefore the data retrieved must already be a boolean. The column ordinal.

### GetByte

```
[C#]          public          byte          GetByte(int          i);
[C++]        public:    __sealed    unsigned    char    GetByte(int    i);
[VB] NotOverridable Public Function GetByte(ByVal i As Integer) As Byte
[JScript]    public    function    GetByte(i    :    int)    :    Byte;
```

## Description

Returns the value of the specified column as a byte.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a byte. The column ordinal.

### GetBytes

[C#] public long GetBytes(int i, long dataIndex, byte[] buffer, int bufferIndex, int length);

[C++] public: \_\_sealed \_\_int64 GetBytes(int i, \_\_int64 dataIndex, unsigned char buffer \_\_gc[], int bufferIndex, int length);

[VB] NotOverridable Public Function GetBytes(ByVal i As Integer, ByVal dataIndex As Long, ByVal buffer() As Byte, ByVal bufferIndex As Integer, ByVal length As Integer) As Long

[JScript] public function GetBytes(i : int, dataIndex : long, buffer : Byte[], bufferIndex : int, length : int) : long;

### Description

Returns the value of the specified column as a byte array.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a byte array. column ordinal. point to start from within the source data. buffer to copy data into. point to start from within the buffer. max length to copy into the buffer.

### GetChar

```

1
2 [C#]          public          char          GetChar(int          i);
3 [C++]        public:          __sealed          __wchar_t          GetChar(int          i);
4 [VB] NotOverridable Public Function GetChar(ByVal i As Integer) As Char
5 [JScript]    public          function          GetChar(i          :          int)          :          Char;
6

```

### *Description*

Returns the value of the specified column as a character.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a character. The column ordinal.

### *GetChars*

```

14 [C#] public long GetChars(int i, long dataIndex, char[] buffer, int bufferIndex, int
15 length);
16 [C++] public: __sealed __int64 GetChars(int i, __int64 dataIndex, __wchar_t
17 buffer          __gc[],          int          bufferIndex,          int          length);
18 [VB] NotOverridable Public Function GetChars(ByVal i As Integer, ByVal
19 dataIndex As Long, ByVal buffer() As Char, ByVal bufferIndex As Integer,
20 ByVal          length          As          Integer)          As          Long
21 [JScript] public function GetChars(i : int, dataIndex : long, buffer : Char[],
22 bufferIndex          :          int,          length          :          .int)          :          long;
23

```

### *Description*

Returns the value of the specified column as a character array.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a character array. column ordinal. point to start from within the source data. buffer to copy data into. point to start from within the buffer. max length to copy into the buffer.

#### GetData

[C#] public IDataReader GetData(int i);

[C++] public: \_\_sealed IDataReader\* GetData(int i);

[VB] NotOverridable Public Function GetData(ByVal i As Integer) As  
IDataReader

[JScript] public function GetData(i : int) : IDataReader;

#### Description

Not currently supported.

#### GetDataTypeName

[C#] public string GetDataTypeName(int i);

[C++] public: \_\_sealed String\* GetDataTypeName(int i);

[VB] NotOverridable Public Function GetDataTypeName(ByVal i As Integer) As  
String

[JScript] public function GetDataTypeName(i : int) : String;

#### Description

1 Returns the name of the back-end data type.

2 *Return Value:* The name of the back-end data type. The column ordinal.

3 GetDateTime

4

5 [C#] public DateTime GetDateTime(int i);

6 [C++] public: \_\_sealed DateTime GetDateTime(int i);

7 [VB] NotOverridable Public Function GetDateTime(ByVal i As Integer) As

8 DateTime

9 [JScript] public function GetDateTime(i : int) : DateTime;

10

11 *Description*

12 Returns the value of the specified column as a **System.DateTime** object.

13 *Return Value:* The value of the specified column.

14 No conversions are performed, therefore the data retrieved must already be

15 a **System.DateTime** object. The column ordinal.

16 GetDecimal

17

18 [C#] public decimal GetDecimal(int i);

19 [C++] public: \_\_sealed Decimal GetDecimal(int i);

20 [VB] NotOverridable Public Function GetDecimal(ByVal i As Integer) As

21 Decimal

22 [JScript] public function GetDecimal(i : int) : Decimal;

23

24 *Description*

25

1 Returns the value of the specified column as a **System.Decimal** object.

2 *Return Value:* The value of the specified column.

3 No conversions are performed, therefore the data retrieved must already be  
4 a **System.Decimal** object. The column ordinal.

5 **GetDouble**

6  
7 [C#] public double GetDouble(int i);

8 [C++] public: \_\_sealed double GetDouble(int i);

9 [VB] NotOverridable Public Function GetDouble(ByVal i As Integer) As Double

10 [JScript] public function GetDouble(i : int) : double;

11  
12 *Description*

13 Returns the value of the specified column as a double-precision floating  
14 point number.

15 *Return Value:* The value of the specified column.

16 No conversions are performed, therefore the data retrieved must already be  
17 a double-precision floating point number. The column ordinal.

18 **GetFieldType**

19  
20 [C#] public Type GetFieldType(int i);

21 [C++] public: \_\_sealed Type\* GetFieldType(int i);

22 [VB] NotOverridable Public Function GetFieldType(ByVal i As Integer) As Type

23 [JScript] public function GetFieldType(i : int) : Type;

24  
25 *Description*

1 Returns the **System.Type** that is the data type of the object.  
 2 *Return Value:* The **System.Type** that is the data type of the object. The column  
 3 ordinal.

4 **GetFloat**

5  
 6 [C#] public float GetFloat(int i);  
 7 [C++] public: \_\_sealed float GetFloat(int i);  
 8 [VB] NotOverridable Public Function GetFloat(ByVal i As Integer) As Single  
 9 [JScript] public function GetFloat(i : int) : float;

10  
 11 *Description*

12 Returns the value of the specified column as a single-precision floating  
 13 point number.

14 *Return Value:* The value of the specified column.

15 No conversions are performed, therefore the data retrieved must already be  
 16 a single-precision floating point number. The column ordinal.

17 **GetGuid**

18  
 19 [C#] public Guid GetGuid(int i);  
 20 [C++] public: \_\_sealed Guid GetGuid(int i);  
 21 [VB] NotOverridable Public Function GetGuid(ByVal i As Integer) As Guid  
 22 [JScript] public function GetGuid(i : int) : Guid;

23  
 24 *Description*  
 25

Returns the guid value of the specified field.

*Return Value:* The guid value of the specified field. The index of the field to find.

GetInt16

[C#] public short GetInt16(int i);

[C++] public: \_\_sealed short GetInt16(int i);

[VB] NotOverridable Public Function GetInt16(ByVal i As Integer) As Short

[JScript] public function GetInt16(i : int) : Int16;

#### *Description*

Returns the value of the specified column as a 16-bit signed integer.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a 16-bit signed integer. The column ordinal.

GetInt32

[C#] public int GetInt32(int i);

[C++] public: \_\_sealed int GetInt32(int i);

[VB] NotOverridable Public Function GetInt32(ByVal i As Integer) As Integer

[JScript] public function GetInt32(i : int) : int;

#### *Description*

Returns the value of the specified column as a 32-bit signed integer.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a 32-bit signed integer. The column ordinal.

### GetInt64

```
[C#]          public          long          GetInt64(int          i);
[C++]         public:         __sealed      __int64      GetInt64(int          i);
[VB] NotOverridable Public Function GetInt64(ByVal i As Integer) As Long
[JScript]     public         function      GetInt64(i      :      int)      :      long;
```

#### Description

Returns the value of the specified column as a 64-bit signed integer.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a 64-bit signed integer. The column ordinal.

### GetName

```
[C#]          public          string          GetName(int          i);
[C++]         public:         __sealed      String*      GetName(int          i);
[VB] NotOverridable Public Function GetName(ByVal i As Integer) As String
[JScript]     public         function      GetName(i      :      int)      :      String;
```

#### Description

Returns the name of the specified column.

*Return Value:* The name of the specified column. The column ordinal.

### GetOrdinal



```

1  [C++]      public:      __sealed      Object*      GetValue(int      i);
2  [VB] NotOverridable Public Function GetValue(ByVal i As Integer) As Object
3  [JScript]  public      function      GetValue(i      :      int)      :      Object;
4

```

#### Description

Returns the value at the specified column in its native format. The column ordinal.

#### GetValues

```

10 [C#]      public      int      GetValues(object[]      values);
11 [C++]      public:      __sealed      int      GetValues(Object*      values      __gc[]);
12 [VB] NotOverridable Public Function GetValues(ByVal values() As Object) As
13 Integer
14 [JScript]  public      function      GetValues(values      :      Object[])      :      int;
15

```

#### Description

Returns all the attribute fields in the collection for the current record.

*Return Value:* The number of instances of **System.Object** in the array.

Using this method may be more effeciant for most applications then retrieving each field individually. An array of **System.Object** to copy the attribute fields into.

#### IsDBNull

```

24 [C#]      public      bool      IsDBNull(int      i);
25 [C++]      public:      __sealed      bool      IsDBNull(int      i);

```

```

1 [VB] NotOverridable Public Function IsDBNull(ByVal i As Integer) As Boolean
2 [JScript]    public    function    IsDBNull(i    :    int)    :    Boolean;

```

3

#### 4 *Description*

5       Used       to       indicate       non-existent       values.

6 *Return Value:* **true** if the specified column is equivalent to **System.DBNull** ;  
7 otherwise, **false** . The column ordinal.

8       ICustomTypeDescriptor.GetAttributes

9

10 [C#]       AttributeCollection       ICustomTypeDescriptor.GetAttributes();

11 [C++]       AttributeCollection\*       ICustomTypeDescriptor::GetAttributes();

12 [VB]       Function       GetAttributes()       As       AttributeCollection       Implements  
13 ICustomTypeDescriptor.GetAttributes

14 [JScript] function ICustomTypeDescriptor.GetAttributes() : AttributeCollection;

15       ICustomTypeDescriptor.GetClassName

16

17 [C#]       string       ICustomTypeDescriptor.GetClassName();

18 [C++]       String\*       ICustomTypeDescriptor::GetClassName();

19 [VB]       Function       GetClassName()       As       String       Implements  
20 ICustomTypeDescriptor.GetClassName

21 [JScript] function ICustomTypeDescriptor.GetClassName() : String;

22       ICustomTypeDescriptor.GetComponentName

23

24 [C#]       string       ICustomTypeDescriptor.GetComponentName();

25 [C++]       String\*       ICustomTypeDescriptor::GetComponentName();

```

1  [VB]    Function    GetComponentName()    As    String    Implements
2  ICustomTypeDescriptor.GetComponentName
3  [JScript] function ICustomTypeDescriptor.GetComponentName() : String;
4          ICustomTypeDescriptor.GetConverter
5
6  [C#]          TypeConverter          ICustomTypeDescriptor.GetConverter();
7  [C++]          TypeConverter*          ICustomTypeDescriptor::GetConverter();
8  [VB]    Function    GetConverter()    As    TypeConverter    Implements
9  ICustomTypeDescriptor.GetConverter
10 [JScript] function ICustomTypeDescriptor.GetConverter() : TypeConverter;
11          ICustomTypeDescriptor.GetDefaultEvent
12
13 [C#]          EventDescriptor          ICustomTypeDescriptor.GetDefaultEvent();
14 [C++]          EventDescriptor*          ICustomTypeDescriptor::GetDefaultEvent();
15 [VB]    Function    GetDefaultEvent()    As    EventDescriptor    Implements
16 ICustomTypeDescriptor.GetDefaultEvent
17 [JScript] function ICustomTypeDescriptor.GetDefaultEvent() : EventDescriptor;
18          ICustomTypeDescriptor.GetDefaultProperty
19
20 [C#]          PropertyDescriptor          ICustomTypeDescriptor.GetDefaultProperty();
21 [C++]          PropertyDescriptor*          ICustomTypeDescriptor::GetDefaultProperty();
22 [VB]    Function    GetDefaultProperty()    As    PropertyDescriptor    Implements
23 ICustomTypeDescriptor.GetDefaultProperty
24 [JScript]    function    ICustomTypeDescriptor.GetDefaultProperty()    :
25 PropertyDescriptor;

```

```

1      ICustomTypeDescriptor.GetEditor
2
3  [C#]    object    ICustomTypeDescriptor.GetEditor(Type    editorBaseType);
4  [C++]   Object*   ICustomTypeDescriptor::GetEditor(Type*   editorBaseType);
5  [VB]   Function GetEditor(ByVal editorBaseType As Type) As Object Implements
6  ICustomTypeDescriptor.GetEditor
7  [JScript] function ICustomTypeDescriptor.GetEditor(editorBaseType : Type) :
8  Object;
9
10     ICustomTypeDescriptor.GetEvents
11
12  [C#]    EventDescriptorCollection    ICustomTypeDescriptor.GetEvents();
13  [C++]   EventDescriptorCollection*    ICustomTypeDescriptor::GetEvents();
14  [VB]   Function GetEvents() As EventDescriptorCollection Implements
15  ICustomTypeDescriptor.GetEvents
16  [JScript] function ICustomTypeDescriptor.GetEvents() :
17  EventDescriptorCollection;
18
19     ICustomTypeDescriptor.GetEvents
20
21  [C#]    EventDescriptorCollection    ICustomTypeDescriptor.GetEvents(Attribute[]
22  attributes);
23  [C++]   EventDescriptorCollection*    ICustomTypeDescriptor::GetEvents(Attribute*
24  attributes[]);
25  [VB]   Function GetEvents(ByVal attributes() As Attribute) As
26  EventDescriptorCollection Implements ICustomTypeDescriptor.GetEvents

```

```

1  [JScript] function ICustomTypeDescriptor.GetEvents(attributes : Attribute[]) :
2  EventDescriptorCollection;
3      ICustomTypeDescriptor.GetProperties
4
5  [C#]    PropertyDescriptorCollection    ICustomTypeDescriptor.GetProperties();
6  [C++]  PropertyDescriptorCollection*    ICustomTypeDescriptor::GetProperties();
7  [VB]   Function  GetProperties() As PropertyDescriptorCollection Implements
8  ICustomTypeDescriptor.GetProperties
9  [JScript]      function      ICustomTypeDescriptor.GetProperties()      :
10 PropertyDescriptorCollection;
11      ICustomTypeDescriptor.GetProperties
12
13 [C#]                                PropertyDescriptorCollection
14 ICustomTypeDescriptor.GetProperties(Attribute[]      attributes);
15 [C++]                                PropertyDescriptorCollection*
16 ICustomTypeDescriptor::GetProperties(Attribute*      attributes[]);
17 [VB]   Function  GetProperties(ByVal  attributes() As  Attribute) As
18 PropertyDescriptorCollection Implements ICustomTypeDescriptor.GetProperties
19 [JScript] function ICustomTypeDescriptor.GetProperties(attributes : Attribute[]) :
20 PropertyDescriptorCollection;
21      ICustomTypeDescriptor.GetPropertyOwner
22
23 [C#]  object ICustomTypeDescriptor.GetPropertyOwner(PropertyDescriptor pd);
24 [C++]  Object* ICustomTypeDescriptor::GetPropertyOwner(PropertyDescriptor*
25 pd);

```

```

1  [VB] Function GetPropertyOwner(ByVal pd As PropertyDescriptor) As Object
2  Implements                                     ICustomPropertyDescriptor.GetPropertyOwner
3  [JScript]      function      ICustomPropertyDescriptor.GetPropertyOwner(pd      :
4  PropertyDescriptor) : Object;
5
6      DbEnumerator class (System.Data.Common)
7
8      ToString
9
10
11
12
13
14  [C#]      public      DbEnumerator(IDataReader      reader);
15  [C++]      public:      DbEnumerator(IDataReader*      reader);
16  [VB]      Public      Sub      New(ByVal      reader      As      IDataReader)
17  [JScript] public function DbEnumerator(reader : IDataReader);
18
19      Current
20
21      ToString
22
23
24
25
26  [C#]      public      object      Current      {get;}
27  [C++]      public:      __property      Object*      get_Current();
28  [VB]      Public      ReadOnly      Property      Current      As      Object
29  [JScript] public      function      get      Current()      :      Object;

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

MoveNext

```
[C#]          public          bool          MoveNext();
[C++]        public:          __sealed        bool          MoveNext();
[VB]  NotOverridable  Public  Function  MoveNext()  As  Boolean
[JScript]    public      function  MoveNext()      :      Boolean;
```

*Description*

Reset

```
[C#]          public          void          Reset();
[C++]        public:          __sealed        void          Reset();
[VB]          NotOverridable          Public          Sub          Reset()
[JScript]          public          function          Reset();
```

*Description*

RowUpdatedEventArgs class (System.Data.Common)  
ToString

*Description*

Provides data for the **RowUpdated** event of a .NET data provider.

The **RowUpdated** event message is typically raised when an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to a row is completed.

**RowUpdatedEventArgs**

*Example Syntax:*

**ToString**

[C#] protected **RowUpdatedEventArgs**(DataRow **dataRow**, IDbCommand **command**, StatementType **statementType**, DataTableMapping **tableMapping**);

[C++] protected: **RowUpdatedEventArgs**(DataRow\* **dataRow**, IDbCommand\* **command**, StatementType **statementType**, DataTableMapping\* **tableMapping**);

[VB] Protected Sub New(ByVal **dataRow** As DataRow, ByVal **command** As IDbCommand, ByVal **statementType** As StatementType, ByVal **tableMapping** As DataTableMapping)

[JScript] protected function **RowUpdatedEventArgs**(**dataRow** : DataRow, **command** : IDbCommand, **statementType** : StatementType, **tableMapping** : DataTableMapping);

### *Description*

Initializes a new instance of the **System.Data.Common.RowUpdatedEventArgs** class. The **System.Data.DataRow** sent through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**. The **System.Data.IDbCommand** executed when **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** is

1 called. The type of SQL statement executed. The  
 2 **System.Data.Common.DataTableMapping** sent through an  
 3 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet).**

4 Command

5 ToString

6  
 7 [C#] public IDbCommand Command {get;}

8 [C++] public: \_\_property IDbCommand\* get\_Command();

9 [VB] Public ReadOnly Property Command As IDbCommand

10 [JScript] public function get Command() : IDbCommand;

11  
 12 *Description*

13 Gets the **System.Data.IDbCommand** executed when  
 14 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** is  
 15 called.

16 Errors

17 ToString

18  
 19 [C#] public Exception Errors {get; set;}

20 [C++] public: \_\_property Exception\* get\_Errors();public: \_\_property void  
 21 set\_Errors(Exception\*);

22 [VB] Public Property Errors As Exception

23 [JScript] public function get Errors() : Exception;public function set  
 24 Errors(Exception);

25

*Description*

Gets any errors generated by the .NET data provider when the **System.Data.Common.RowUpdatedEventArgs.Command** was executed.

RecordsAffected  
ToString

```
[C#]          public          int          RecordsAffected          {get;}
[C++]          public:          __property          int          get_RecordsAffected();
[VB]    Public    ReadOnly    Property    RecordsAffected    As    Integer
[JScript]    public    function    get    RecordsAffected()    :    int;
```

*Description*

Gets the number of rows changed, inserted, or deleted by execution of the SQL statement.

Row  
ToString

```
[C#]          public          DataRow          Row          {get;}
[C++]          public:          __property          DataRow*          get_Row();
[VB]    Public    ReadOnly    Property    Row    As    DataRow
[JScript]    public    function    get    Row()    :    DataRow;
```

*Description*

Gets the **System.Data.DataRow** sent through an  
**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** .

StatementType

ToString

[C#]        public        StatementType        StatementType        {get;}

[C++]    public:    \_\_property    StatementType    get\_StatementType();

[VB]    Public    ReadOnly    Property    StatementType    As    StatementType

[JScript]    public    function    get    StatementType()    :    StatementType;

#### *Description*

Gets the type of SQL statement executed.

**System.Data.Common.RowUpdatedEventArgs.StatementType** can be  
one of the following values: Select Insert Update Delete

Status

ToString

[C#]        public        UpdateStatus        Status        {get;        set;}

[C++]    public:    \_\_property    UpdateStatus    get\_Status();public:    \_\_property    void  
set\_Status(UpdateStatus);

[VB]        Public        Property        Status        As        UpdateStatus

[JScript]    public    function    get    Status()    :    UpdateStatus;public    function    set  
Status(UpdateStatus);

#### *Description*

1 Gets the **System.Data.UpdateStatus** of the  
2 **System.Data.Common.RowUpdatedEventArgs.Command** .

3 TableMapping

4 ToString

5  
6 [C#] public DataTableMapping TableMapping {get;}

7 [C++] public: \_\_property DataTableMapping\* get\_TableMapping();

8 [VB] Public ReadOnly Property TableMapping As DataTableMapping

9 [JScript] public function get TableMapping() : DataTableMapping;

10  
11 *Description*

12 Gets the **System.Data.Common.DataTableMapping** sent through an  
13 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** .

14 RowUpdatingEventArgs class (System.Data.Common)

15 ToString

16  
17  
18 *Description*

19 Provides the data for the **RowUpdating** event of a .NET data provider.

20 The **RowUpdating** event is typically raised just before an  
21 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to a  
22 row begins.

23 RowUpdatingEventArgs

24 *Example Syntax:*

25 ToString

```
[C#] protected RowUpdatingEventArgs(DataRow dataRow, IDbCommand
command, StatementType statementType, DataTableMapping tableMapping);
[C++] protected: RowUpdatingEventArgs(DataRow* dataRow, IDbCommand*
command, StatementType statementType, DataTableMapping* tableMapping);
[VB] Protected Sub New(ByVal dataRow As DataRow, ByVal command As
IDbCommand, ByVal statementType As StatementType, ByVal tableMapping As
DataTableMapping)
[JavaScript] protected function RowUpdatingEventArgs(dataRow : DataRow,
command : IDbCommand, statementType : StatementType, tableMapping :
DataTableMapping);
```

### Description

Initializes a new instance of the **System.Data.Common.RowUpdatingEventArgs** class. The **System.Data.DataRow** to **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet).** The **System.Data.IDbCommand** to execute when **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** is called. The type of SQL statement to execute. The **System.Data.Common.DataTableMapping** to send through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet).**

## Command

ToString

```

1
2 [C#]      public      IDbCommand      Command      {get;      set;}
3 [C++] public: __property IDbCommand* get_Command();public: __property void
4 set_Command(IDbCommand*);
5 [VB]      Public      Property      Command      As      IDbCommand
6 [JScript] public function get Command() : IDbCommand;public function set
7 Command(IDbCommand);

```

### *Description*

Gets the **System.Data.IDbCommand** to execute during the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** operation.

Errors

ToString

```

16 [C#]      public      Exception      Errors      {get;      set;}
17 [C++] public: __property Exception* get_Errors();public: __property void
18 set_Errors(Exception*);
19 [VB]      Public      Property      Errors      As      Exception
20 [JScript] public function get Errors() : Exception;public function set
21 Errors(Exception);

```

### *Description*

Gets any errors generated by the .NET data provider when the **System.Data.Common.RowUpdatedEventArgs.Command** executes.

1	Row
2	ToString
3	
4	[C#] public DataRow Row {get;}
5	[C++] public: __property DataRow* get_Row();
6	[VB] Public ReadOnly Property Row As DataRow
7	[JScript] public function get Row() : DataRow;
8	
9	<i>Description</i>
10	Gets the <b>System.Data.DataRow</b> to send through an
11	<b>System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)</b> .
12	StatementType
13	ToString
14	
15	[C#] public StatementType StatementType {get;}
16	[C++] public: __property StatementType get_StatementType();
17	[VB] Public ReadOnly Property StatementType As StatementType
18	[JScript] public function get StatementType() : StatementType;
19	
20	<i>Description</i>
21	Gets the type of SQL statement to execute.
22	<b>System.Data.Common.RowUpdatingEventArgs.StatementType</b> can be
23	one of the following values: Select Insert Update Delete Indicates the type of SQL
24	command to execute. This property is read-only.
25	Status

ToString

```
[C#]      public      UpdateStatus      Status      {get;      set;}
[C++] public: __property UpdateStatus get_Status();public: __property void
set_Status(UpdateStatus);
[VB]      Public      Property      Status      As      UpdateStatus
[JavaScript] public function get Status() : UpdateStatus;public function set
Status(UpdateStatus);
```

#### *Description*

Gets the **System.Data.UpdateStatus** of the

### **System.Data.OleDb**

#### *Description*

The **System.Data.OleDb** namespace is the OLE DB .NET Data Provider.

OleDbCommand class (System.Data.OleDb)

#### *Description*

Represents a SQL statement or stored procedure to execute at a data source.

When an instance of **System.Data.OleDb.OleDbCommand** is created, the read/write properties are set to their initial values. For a list of these values, see the **System.Data.OleDb.OleDbCommand** constructor.

Constructors:

OleDbCommand

*Example Syntax:*

[C#] public OleDbCommand();

[C++] public: OleDbCommand();

[VB] Public Sub New()

[JScript] public function OleDbCommand(); Initializes a new instance of the **System.Data.OleDb.OleDbCommand** class.

#### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbCommand** class.

The following table shows initial property values for an instance of **System.Data.OleDb.OleDbCommand**.

OleDbCommand

*Example Syntax:*

[C#] public OleDbCommand(string cmdText);

[C++] public: OleDbCommand(String\* cmdText);

[VB] Public Sub New(ByVal cmdText As String)

[JScript] public function OleDbCommand(cmdText : String);

## Description

Initializes a new instance of the **System.Data.OleDb.OleDbCommand** class with the text of the query.

The following table shows initial property values for an instance of **System.Data.OleDb.OleDbCommand** . The text of the query.

OleDbCommand

### Example Syntax:

```
[C#] public OleDbCommand(string cmdText, OleDbConnection connection);
[C++] public: OleDbCommand(String* cmdText, OleDbConnection* connection);
[VB] Public Sub New(ByVal cmdText As String, ByVal connection As
OleDbConnection)
[JScript] public function OleDbCommand(cmdText : String, connection :
OleDbConnection);
```

## Description

Initializes a new instance of the **System.Data.OleDb.OleDbCommand** class with the text of the query and an **System.Data.OleDb.OleDbConnection** .

The following table shows initial property values for an instance of **System.Data.OleDb.OleDbCommand** . The text of the query. An **System.Data.OleDb.OleDbConnection** that represents the connection to a data source.

OleDbCommand

### Example Syntax:

```

[C#] public OleDbCommand(string cmdText, OleDbConnection connection,
OleDbTransaction transaction);

[C++] public: OleDbCommand(String* cmdText, OleDbConnection* connection,
OleDbTransaction* transaction);

[VB] Public Sub New(ByVal cmdText As String, ByVal connection As
OleDbConnection, ByVal transaction As OleDbTransaction)

[JScript] public function OleDbCommand(cmdText : String, connection :
OleDbConnection, transaction : OleDbTransaction);
    
```

### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbCommand** class with the text of the query, an **System.Data.OleDb.OleDbConnection** , and the **System.Data.OleDb.OleDbCommand.Transaction** .

The following table shows initial property values for an instance of **System.Data.OleDb.OleDbCommand** . The text of the query. An **System.Data.OleDb.OleDbConnection** that represents the connection to a data source. The transaction in which the **System.Data.OleDb.OleDbCommand** executes.

Properties:

CommandText

```

[C#] public string CommandText {get; set;}

[C++] public: __property String* get_CommandText();public: __property void
set_CommandText(String*);
    
```

1 [VB] Public Property CommandText As String

2 [JScript] public function get CommandText() : String;public function set

3 CommandText(String);

4  
5 *Description*

6 Gets or sets the SQL statement or stored procedure to execute at the data  
7 source.

8 When the **System.Data.IDbCommand.CommandType** property is set to  
9 **StoredProcedure** , the **System.Data.OleDb.OleDbCommand.CommandText**  
10 property should be set to the name of the stored procedure. The command executes  
11 this stored procedure when you call one of the Execute methods.

12 **CommandTimeout**

13  
14 [C#] public int CommandTimeout {get; set;}

15 [C++] public: \_\_property int get\_CommandTimeout();public: \_\_property void  
16 set\_CommandTimeout(int);

17 [VB] Public Property CommandTimeout As Integer

18 [JScript] public function get CommandTimeout() : int;public function set

19 CommandTimeout(int);

20  
21 *Description*

22 Gets or sets the wait time before terminating an attempt to execute a  
23 command and generating an error.

A value of 0 indicates no limit, and should be avoided in a **System.Data.OleDb.OleDbCommand.CommandTimeout** because an attempt to execute a command will wait indefinitely.

#### CommandType

```
[C#] public CommandType CommandType {get; set;}
[C++] public: __property CommandType get _CommandType();public: __property
void set _CommandType(CommandType);
[VB] Public Property CommandType As CommandType
[JScript] public function get CommandType() : CommandType;public function set
CommandType(CommandType);
```

#### *Description*

Gets or sets a value indicating how the **System.Data.OleDb.OleDbCommand.CommandText** property is interpreted.

When you set the **System.Data.OleDb.OleDbCommand.CommandType** property to **StoredProcedure**, you should set the **System.Data.OleDb.OleDbCommand.CommandText** property to the name of the stored procedure. The command executes this stored procedure when you call one of the Execute methods.

#### Connection

```
[C#] public OleDbConnection Connection {get; set;}
[C++] public: __property OleDbConnection* get _Connection();public: __property
void set _Connection(OleDbConnection*);
```

1 [VB] Public Property Connection As OleDbConnection

2 [JScript] public function get Connection() : OleDbConnection;public function set

3 Connection(OleDbConnection);

4  
5 *Description*

6 Gets or sets the **System.Data.OleDb.OleDbConnection** used by this  
7 instance of the **System.Data.OleDb.OleDbCommand** .

8 You cannot set the **System.Data.OleDb.OleDbCommand.Connection** ,  
9 **System.Data.OleDb.OleDbCommand.CommandType** and  
10 **System.Data.OleDb.OleDbCommand.CommandText** properties if the current  
11 connection is performing an execute or fetch operation.

12 Container

13 DesignMode

14 DesignTimeVisible

15  
16  
17 *Description*

18 Gets or sets a value indicating whether the command object should be  
19 visible in a customized Windows Forms Designer control.

20 Events

21 Parameters

22  
23  
24 *Description*

25 Gets the **System.Data.OleDb.OleDbParameterCollection** .

1       The OLE DB .NET Provider does not support named parameters for  
2       passing parameters to a SQL Statement or a stored procedure called by an  
3       **System.Data.OleDb.OleDbCommand** when  
4       **System.Data.OleDb.OleDbCommand.CommandType** is set to **Text** . In this  
5       case, the question mark (?) placeholder must be used. For example: **SELECT \***  
6       **FROM Customers WHERE CustomerID = ?** As a result, the order in which  
7       **System.Data.OleDb.OleDbParameter** objects are added to the  
8       **System.Data.OleDb.OleDbParameterCollection** must directly correspond to the  
9       position of the question mark placeholder for the parameter.

10       Site

11       Transaction

12  
13  
14       *Description*

15       Gets or sets the transaction in which the  
16       **System.Data.OleDb.OleDbCommand** executes.

17       UpdatedRowSource

18  
19       [C#] public UpdateRowSource UpdatedRowSource {get; set;}

20       [C++] public: \_\_property UpdateRowSource get\_UpdatedRowSource();public:

21       \_\_property void set\_UpdatedRowSource(UpdateRowSource);

22       [VB] Public Property UpdatedRowSource As UpdateRowSource

23       [JScript] public function get UpdatedRowSource() : UpdateRowSource;public

24       function set UpdatedRowSource(UpdateRowSource);

25

1  
2 *Description*

3 Gets or sets how command results are applied to the  
4 **System.Data.DataRow** when used by the  
5 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** method  
6 of the **System.Data.Common.DbDataAdapter** .

7 Methods:

8 Cancel

9  
10 [C#] public void Cancel();

11 [C++] public: \_\_sealed void Cancel();

12 [VB] NotOverridable Public Sub Cancel()

13 [JScript] public function Cancel();

14  
15 *Description*

16 Cancels the execution of an **System.Data.OleDb.OleDbCommand** .

17 If there is nothing to cancel, nothing happens.

18 CreateParameter

19  
20 [C#] public OleDbParameter CreateParameter();

21 [C++] public: OleDbParameter\* CreateParameter();

22 [VB] Public Function CreateParameter() As OleDbParameter

23 [JScript] public function CreateParameter() : OleDbParameter;

24  
25 *Description*

Creates a new instance of an **System.Data.OleDb.OleDbParameter** object.

*Return Value:* An **System.Data.OleDb.OleDbParameter** object.

The **System.Data.OleDb.OleDbCommand.CreateCommand** method is a strongly-typed version of **System.Data.IDbCommand.CreateCommand** .

## Dispose

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean); Releases the resources used by the **System.Data.OleDb.OleDbCommand** .

## Description

Releases the unmanaged resources used by the **System.Data.OleDb.OleDbCommand** and optionally releases the managed resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

## ExecuteNonQuery

[C#] public int ExecuteNonQuery();

[C++] public: \_\_sealed int ExecuteNonQuery();

[VB] NotOverridable Public Function ExecuteNonQuery() As Integer

1 [JScript] public function ExecuteNonQuery() : int;

3 *Description*

4 Executes a SQL statement against the  
5 **System.Data.OleDb.OleDbCommand.Connection** and returns the number of  
6 rows affected.

7 *Return Value:* The number of rows affected.

8 You can use the  
9 **System.Data.SqlClient.SqlCommand.ExecuteNonQuery** to perform catalog  
10 operations (for example, querying the structure of a database or creating database  
11 objects such as tables), or to change the data in a database without using a  
12 **System.Data.DataSet** by executing UPDATE, INSERT, or DELETE statements.

13 **ExecuteReader**

15 [C#] public OleDbDataReader ExecuteReader();

16 [C++] public: OleDbDataReader\* ExecuteReader();

17 [VB] Public Function ExecuteReader() As OleDbDataReader

18 [JScript] public function ExecuteReader() : OleDbDataReader; Sends the

19 **System.Data.OleDb.OleDbCommand.CommandText** to the

20 **System.Data.OleDb.OleDbCommand.Connection** and builds an

21 **System.Data.OleDb.OleDbDataReader** .

23 *Description*

24 Sends the **System.Data.OleDb.OleDbCommand.CommandText** to the  
25 **System.Data.OleDb.OleDbCommand.Connection** and builds an

## **System.Data.OleDb.OleDbDataReader .**

*Return Value:* An **System.Data.OleDb.OleDbDataReader** object.

When the **System.Data.IDbCommand.CommandType** property is set to **StoredProcedure** , the **System.Data.OleDb.OleDbCommand.CommandText** property should be set to the name of the stored procedure. The command executes this stored procedure when you call

## **System.Data.OleDb.OleDbCommand.ExecuteReader .**

**ExecuteReader**

```
[C#] public OleDbDataReader ExecuteReader(CommandBehavior behavior);  
[C++] public: OleDbDataReader* ExecuteReader(CommandBehavior behavior);  
[VB] Public Function ExecuteReader(ByVal behavior As CommandBehavior) As  
OleDbDataReader  
[JScript] public function ExecuteReader(behavior : CommandBehavior) :  
OleDbDataReader;
```

## *Description*

Sends the **System.Data.OleDb.OleDbCommand.CommandText** to the **System.Data.OleDb.OleDbCommand.Connection** , and builds an **System.Data.OleDb.OleDbDataReader** using one of the **System.Data.CommandBehavior** values.

*Return Value:* An **System.Data.OleDb.OleDbDataReader** object.

When you specify **System.Data.CommandBehavior.SingleRow** with the **System.Data.OleDb.OleDbCommand.ExecuteReader** method of the **System.Data.OleDb.OleDbCommand** object, the OLE DB .NET Data Provider

performs binding using the OLE DB IRow interface if it is available. Otherwise, it uses the IRowset interface. If your SQL statement is expected to return only a single row, specifying **System.Data.CommandBehavior.SingleRow** can also improve application performance. One of the **System.Data.CommandBehavior** values.

#### ExecuteScalar

```
[C#] public object ExecuteScalar();
[C++] public: __sealed Object* ExecuteScalar();
[VB] NotOverridable Public Function ExecuteScalar() As Object
[JScript] public function ExecuteScalar() : Object;
```

#### *Description*

Executes the query, and returns the first column of the first row in the resultset returned by the query. Extra columns or rows are ignored.

**Return Value:** The first column of the first row in the resultset.

Use the **System.Data.OleDb.OleDbCommand.ExecuteScalar** method to retrieve a single value (for example, an aggregate value) from a data source. This requires less code than using the **System.Data.OleDb.OleDbCommand.ExecuteReader** method, and then performing the operations necessary to generate the single value using the data returned by an **System.Data.OleDb.OleDbDataReader**.

#### Prepare

```
[C#] public void Prepare();
```

1 [C++] public: \_\_sealed void Prepare();

2 [VB] NotOverridable Public Sub Prepare()

3 [JScript] public function Prepare();

4  
5 *Description*

6 Creates a prepared (or compiled) version of the command on the data  
7 source.

8 If the **System.Data.OleDb.OleDbCommand.CommandType** property is  
9 set to **TableDirect** , **System.Data.OleDb.OleDbCommand.Prepare** does  
10 nothing. If **System.Data.OleDb.OleDbCommand.CommandType** is set to  
11 **StoredProcedure** , the call to **System.Data.OleDb.OleDbCommand.Prepare**  
12 should succeed, although it may result in a no-op.

13 **ResetCommandTimeout**

14  
15 [C#] public void ResetCommandTimeout();

16 [C++] public: void ResetCommandTimeout();

17 [VB] Public Sub ResetCommandTimeout()

18 [JScript] public function ResetCommandTimeout();

19  
20 *Description*

21 Resets the **System.Data.OleDb.OleDbCommand.CommandTimeout**  
22 property to the default value.

23 The default value of the  
24 **System.Data.OleDb.OleDbCommand.CommandTimeout** is 30 seconds.

25 **IDbCommand.CreateParameter**

```

1
2 [C#] IDbDataParameter IDbCommand.CreateParameter();
3 [C++] IDbDataParameter* IDbCommand::CreateParameter();
4 [VB] Function CreateParameter() As IDbDataParameter Implements
5     IDbCommand.CreateParameter
6 [JScript] function IDbCommand.CreateParameter() : IDbDataParameter;
7     IDbCommand.ExecuteReader
8
9 [C#] IDataReader IDbCommand.ExecuteReader();
10 [C++] IDataReader* IDbCommand::ExecuteReader();
11 [VB] Function ExecuteReader() As IDataReader Implements
12     IDbCommand.ExecuteReader
13 [JScript] function IDbCommand.ExecuteReader() : IDataReader;
14     IDbCommand.ExecuteReader
15
16 [C#] IDataReader IDbCommand.ExecuteReader(CommandBehavior behavior);
17 [C++] IDataReader* IDbCommand::ExecuteReader(CommandBehavior
18     behavior);
19 [VB] Function ExecuteReader(ByVal behavior As CommandBehavior) As
20     IDataReader Implements IDbCommand.ExecuteReader
21 [JScript] function IDbCommand.ExecuteReader(behavior : CommandBehavior) :
22     IDataReader;
23     ICloneable.Clone
24
25 [C#] object ICloneable.Clone();

```

1 [C++] Object\* ICloneable::Clone();  
 2 [VB] Function Clone() As Object Implements ICloneable.Clone  
 3 [JScript] function ICloneable.Clone() : Object;

4 OleDbCommandBuilder class (System.Data.OleDb)  
 5 ToString

8 *Description*

9 Provides a means of automatically generating single-table commands used  
 10 to reconcile changes made to a **System.Data.DataSet** with the associated  
 11 database. This class cannot be inherited.

12 The **System.Data.OleDb.OleDbDataAdapter** does not automatically  
 13 generate the SQL statements required to reconcile changes made to a  
 14 **System.Data.DataSet** with the associated data source. However, you can create  
 15 an **System.Data.OleDb.OleDbCommandBuilder** object to automatically  
 16 generate SQL statements for single-table updates if you set the  
 17 **System.Data.OleDb.OleDbDataAdapter.SelectCommand** property of the  
 18 **System.Data.OleDb.OleDbDataAdapter** . Then, any additional SQL statements  
 19 that you do not set are generated by the  
 20 **System.Data.OleDb.OleDbCommandBuilder** .

21 OleDbCommandBuilder

22 *Example Syntax:*

23 ToString

25 [C#] public OleDbCommandBuilder();

[C++] public: OleDbCommandBuilder();

[VB] Public Sub New()

[JScript] public function OleDbCommandBuilder(); Initializes a new instance of the **System.Data.OleDb.OleDbCommandBuilder** class.

*Description*

Initializes a new instance of the **System.Data.OleDb.OleDbCommandBuilder** class.

OleDbCommandBuilder

*Example Syntax:*

ToString

[C#] public OleDbCommandBuilder(OleDbDataAdapter adapter);

[C++] public: OleDbCommandBuilder(OleDbDataAdapter\* adapter);

[VB] Public Sub New(ByVal adapter As OleDbDataAdapter)

[JScript] public function OleDbCommandBuilder(adapter : OleDbDataAdapter);

*Description*

Initializes a new instance of the **System.Data.OleDb.OleDbCommandBuilder** class with the associated **System.Data.OleDb.OleDbDataAdapter** object. An **System.Data.OleDb.OleDbDataAdapter**.

Container

DataAdapter

ToString

### *Description*

Gets or sets an **System.Data.OleDb.OleDbDataAdapter** object for which SQL statements are automatically generated.

The **System.Data.OleDb.OleDbCommandBuilder** registers itself as a listener for **System.Data.OleDb.OleDbDataAdapter.RowUpdating** events generated by the **System.Data.OleDb.OleDbDataAdapter**.

**DesignMode**

**Events**

**QuotePrefix**

**ToString**

### *Description*

Gets or sets the beginning character or characters to use when specifying database object names, (for example, tables or columns), that contain characters such as spaces.

Some data sources may have objects that can contain characters such as spaces, commas, and semicolons. To accommodate this capability, use the **System.Data.OleDb.OleDbCommandBuilder.QuotePrefix** and **System.Data.OleDb.OleDbCommandBuilder.QuoteSuffix** properties to specify delimiters such as a left bracket and a right bracket to encapsulate the object name.

**QuoteSuffix**

**ToString**

[C#] public string QuoteSuffix {get; set;}

[C++] public: \_\_property String\* get\_QuoteSuffix();public: \_\_property void  
set\_QuoteSuffix(String\*);

[VB] Public Property QuoteSuffix As String

[JScript] public function get QuoteSuffix() : String;public function set  
QuoteSuffix(String);

### *Description*

Gets or sets the ending character or characters to use when specifying  
database object names, (for example, tables or columns), that contain characters  
such as spaces.

Some data sources may have objects that can contain characters such as  
spaces, commas, and semicolons. To accommodate this capability, use the  
**System.Data.OleDb.OleDbCommandBuilder.QuotePrefix** and  
**System.Data.OleDb.OleDbCommandBuilder.QuoteSuffix** properties to specify  
delimiters such as a left bracket and a right bracket to encapsulate the object name.

Site

DeriveParameters

[C#] public static void DeriveParameters(OleDbCommand command);

[C++] public: static void DeriveParameters(OleDbCommand\* command);

[VB] Public Shared Sub DeriveParameters(ByVal command As OleDbCommand)

[JScript] public static function DeriveParameters(command : OleDbCommand);

1  
2 *Description*

3       Dispose

4  
5 [C#] protected override void Dispose(bool disposing);

6 [C++] protected: void Dispose(bool disposing);

7 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

8 [JScript] protected override function Dispose(disposing : Boolean); Releases the  
9 resources used by the **System.Data.OleDb.OleDbCommandBuilder** .

10  
11 *Description*

12       Releases the unmanaged resources used by the  
13 **System.Data.OleDb.OleDbCommandBuilder** and optionally releases the  
14 managed resources.

15       This method is called by the public method and the  
16 **System.Object.Finalize** method. **true** to release both managed and unmanaged  
17 resources; **false** to release only unmanaged resources.

18       GetDeleteCommand

19  
20 [C#] public OleDbCommand GetDeleteCommand();

21 [C++] public: OleDbCommand\* GetDeleteCommand();

22 [VB] Public Function GetDeleteCommand() As OleDbCommand

23 [JScript] public function GetDeleteCommand() : OleDbCommand;

24  
25 *Description*

Gets the automatically generated SQL statement required to perform deletions at the data source when an application calls **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** on the **System.Data.OleDb.OleDbDataAdapter** .

*Return Value:* The text of the SQL statement to be executed.

An application can use the **System.Data.OleDb.OleDbCommandBuilder.GetDeleteCommand** method for informational or troubleshooting purposes because it returns the text of the statement to be executed.

#### GetInsertCommand

```
[C#] public OleDbCommand GetInsertCommand();
[C++] public: OleDbCommand* GetInsertCommand();
[VB] Public Function GetInsertCommand() As OleDbCommand
[JScript] public function GetInsertCommand() : OleDbCommand;
```

#### *Description*

Gets the automatically generated SQL statement required to perform inserts at the data source when an application calls **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** on the **System.Data.OleDb.OleDbDataAdapter** .

*Return Value:* The text of the SQL statement to be executed.

An application can use the **System.Data.OleDb.OleDbCommandBuilder.GetInsertCommand** method for

informational or troubleshooting purposes because it returns the text of the statement to be executed.

### GetUpdateCommand

```
[C#] public OleDbCommand GetUpdateCommand();  
[C++] public: OleDbCommand* GetUpdateCommand();  
[VB] Public Function GetUpdateCommand() As OleDbCommand  
[JScript] public function GetUpdateCommand() : OleDbCommand;
```

### *Description*

Gets the automatically generated SQL statement required to perform updates at the data source when an application calls **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** on the **System.Data.OleDb.OleDbDataAdapter**.

*Return Value:* The text of the SQL statement to be executed.

An application can use the **System.Data.OleDb.OleDbCommandBuilder.GetUpdateCommand** method for informational or troubleshooting purposes because it returns the text of the statement to be executed.

### RefreshSchema

```
[C#] public void RefreshSchema();  
[C++] public: void RefreshSchema();  
[VB] Public Sub RefreshSchema()  
[JScript] public function RefreshSchema();
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.

An application should call **System.Data.OleDb.OleDbCommandBuilder.RefreshSchema** whenever the SELECT statement associated with the **System.Data.OleDb.OleDbCommandBuilder** changes.

OleDbConnection class (System.Data.OleDb)  
ToString

*Description*

Represents an open connection to a data source.  
An **System.Data.OleDb.OleDbConnection** object represents a unique connection to a data source. In the case of a client/server database system, it is equivalent to a network connection to the server. Depending on the functionality supported by the native OLE DB provider, some collections, methods, or properties of an **System.Data.OleDb.OleDbConnection** object may not be available.

OleDbConnection  
*Example Syntax:*  
ToString

[C#] public OleDbConnection();

1 [C++] public: OleDbConnection();  
 2 [VB] Public Sub New()  
 3 [JScript] public function OleDbConnection(); Initializes a new instance of the  
 4 **System.Data.OleDb.OleDbConnection** class.

5  
 6 *Description*

7       Initializes a new instance of the **System.Data.OleDb.OleDbConnection**  
 8 class.

9       When a new instance of **System.Data.OleDb.OleDbConnection** is  
 10 created, the read/write properties are set to the following initial values unless they  
 11 are specifically set using their associated keywords in the  
 12 **System.Data.SqlClient.SqlConnection.ConnectionString** property.

13       OleDbConnection

14       *Example Syntax:*

15       ToString

16  
 17 [C#] public OleDbConnection(string connectionString);  
 18 [C++] public: OleDbConnection(String\* connectionString);  
 19 [VB] Public Sub New(ByVal connectionString As String)  
 20 [JScript] public function OleDbConnection(connectionString : String);

21  
 22 *Description*

23       Initializes a new instance of the **System.Data.OleDb.OleDbConnection**  
 24 class with the specified connection string.

25

When a new instance of **System.Data.OleDb.OleDbConnection** is created, the read/write properties are set to the following initial values unless they are specifically set using their associated keywords in the **System.Data.SqlClient.SqlConnection.ConnectionString** property. The connection used to open the database.

ConnectionString

ToString

[C#] public string ConnectionString {get; set;}

[C++] public: \_\_property String\* get\_ConnectionString();public: \_\_property void set\_ConnectionString(String\*);

[VB] Public Property ConnectionString As String

[JScript] public function get ConnectionString() : String;public function set ConnectionString(String);

#### *Description*

Gets or sets the string used to open a database.

The **System.Data.OleDb.OleDbConnection.ConnectionString** is designed to match OLE DB connection string format as closely as possible with the following exceptions: The "Provider = *value*" clause is required. However, you cannot use "Provider = MSDASQL" because the OLE DB .NET Data Provider does not support the OLE DB Provider for ODBC (MSDASQL).

ConnectionTimeout

ToString

```

1
2 [C#] public int ConnectionTimeout {get;}
3 [C++] public: __property int get_ConnectionTimeout();
4 [VB] Public ReadOnly Property ConnectionTimeout As Integer
5 [JScript] public function get ConnectionTimeout() : int;
6

```

### *Description*

Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error.

A value of 0 indicates no limit, and should be avoided in a **System.Data.OleDb.OleDbConnection.ConnectionString** because an attempt to connect will wait indefinitely.

Container

Database

ToString

### *Description*

Gets the name of the current database or the database to be used once a connection is open.

The **System.Data.OleDb.OleDbConnection.Database** property updates dynamically. If you change the current database using a SQL statement or the **System.Data.OleDb.OleDbConnection.ChangeDatabase(System.String)** method, an informational message is sent and the property is updated automatically.

1 DataSource  
 2 ToString  
 3  
 4 [C#] public string DataSource {get;}  
 5 [C++] public: \_\_property String\* get\_DataSource();  
 6 [VB] Public ReadOnly Property DataSource As String  
 7 [JScript] public function get DataSource() : String;

8  
 9 *Description*

10 Gets the location and file name of the data source.

11 DesignMode

12 Events

13 Provider

14 ToString

15  
 16  
 17 *Description*

18 Gets the name of the OLE DB provider.

19 ServerVersion

20 ToString

21  
 22 [C#] public string ServerVersion {get;}  
 23 [C++] public: \_\_property String\* get\_ServerVersion();  
 24 [VB] Public ReadOnly Property ServerVersion As String  
 25 [JScript] public function get ServerVersion() : String;

## Description

Gets a string containing the version of the of the server to which the client is connected.

The **System.Data.OleDb.OleDbConnection.ServerVersion** property maps to the OLE DB DBPROP\_DBMSVER property. If **System.Data.OleDb.OleDbConnection.ServerVersion** is not supported by the underlying OLE DB provider, an empty string is returned.

Site

State

ToString

## Description

Gets the current state of the connection.

The allowed state changes are: From **Closed** to **Open** , using the **Open** method of the connnection object.

ToString

## Description

Occurs when the provider sends a warning or an informational message.

Clients that want to process warnings or informational messages sent by the server should create an **System.Data.OleDb.OleDbInfoMessageEventHandler** delegate to listen to this event.

ToString

```
[C#] public event StateChangeEventHandler StateChange;  
[C++] public: __event StateChangeEventHandler* StateChange;  
[VB] Public Event StateChange As StateChangeEventHandler
```

*Description*

Occurs when the state of the connection changes.

The **System.Data.OleDb.OleDbConnection.StateChange** event fires whenever the **System.Data.OleDb.OleDbConnection.State** changes from closed to opened, or from opened to closed.

BeginTransaction

```
[C#] public OleDbTransaction BeginTransaction();  
[C++] public: OleDbTransaction* BeginTransaction();  
[VB] Public Function BeginTransaction() As OleDbTransaction  
[JScript] public function BeginTransaction() : OleDbTransaction;
```

*Description*

Begins a database transaction.

*Return Value:* An object representing the new transaction.

You must explicitly commit or roll back the transaction using the **System.Data.OleDb.OleDbTransaction.Commit** or **System.Data.OleDb.OleDbTransaction.Rollback** method. To ensure that the OLE DB .NET Data Provider transaction management model performs correctly,

avoid using other transaction management models, such as those provided by the data source.

### BeginTransaction

[C#] public OleDbTransaction BeginTransaction(IsolationLevel isolationLevel);

[C++] public: OleDbTransaction\* BeginTransaction(IsolationLevel isolationLevel);

[VB] Public Function BeginTransaction(ByVal isolationLevel As IsolationLevel) As OleDbTransaction

[JScript] public function BeginTransaction(isolationLevel : IsolationLevel) :

OleDbTransaction; Begins a database transaction.

### Description

Begins a database transaction with the current **System.Data.IsolationLevel** value.

*Return Value:* An object representing the new transaction.

You must explicitly commit or roll back the transaction using the **System.Data.OleDb.OleDbTransaction.Commit** or **System.Data.OleDb.OleDbTransaction.Rollback** method. To ensure that the OLE DB .NET Data Provider transaction management model performs correctly, avoid using other transaction management models, such as those provided by the data source. The transaction isolation level for this connection.

### ChangeDatabase

[C#] public void ChangeDatabase(string value);

```

1 [C++] public: __sealed void ChangeDatabase(String* value);
2 [VB] NotOverridable Public Sub ChangeDatabase(ByVal value As String)
3 [JScript] public function ChangeDatabase(value : String);
4

```

#### Description

Changes the current database for an open

#### **System.Data.OleDb.OleDbConnection .**

The value supplied in the *database* parameter must be a valid database name. The *database* parameter cannot contain a null value, be empty, or contain a string with only blank characters. The database name.

#### Close

```

13 [C#] public void Close();
14 [C++] public: __sealed void Close();
15 [VB] NotOverridable Public Sub Close()
16 [JScript] public function Close();
17

```

#### Description

Closes the connection to the data source. This is the preferred method of closing any open connection.

The **System.Data.OleDb.OleDbConnection.Close** method rolls back any pending transactions. It then releases the connection to the connection pool, or closes the connection if connection pooling is disabled. If **System.Data.OleDb.OleDbConnection.Close** is called while handling a

1 **System.Data.OleDb.OleDbConnection.StateChange** event, no additional

2 **System.Data.OleDb.OleDbConnection.StateChange** events are fired.

3 **CreateCommand**

4  
5 [C#] public OleDbCommand CreateCommand();

6 [C++] public: OleDbCommand\* CreateCommand();

7 [VB] Public Function CreateCommand() As OleDbCommand

8 [JScript] public function CreateCommand() : OleDbCommand;

9  
10 *Description*

11 Creates and returns an **System.Data.OleDb.OleDbCommand** object  
12 associated with the **System.Data.OleDb.OleDbConnection** .

13 *Return Value:* An **System.Data.OleDb.OleDbCommand** object.

14 **Dispose**

15  
16 [C#] protected override void Dispose(bool disposing);

17 [C++] protected: void Dispose(bool disposing);

18 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

19 [JScript] protected override function Dispose(disposing : Boolean); Releases the  
20 resources used by the **System.Data.OleDb.OleDbConnection** .

21  
22 *Description*

23 Releases the unmanaged resources used by the  
24 **System.Data.OleDb.OleDbConnection** and optionally releases the managed  
25 resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

#### GetOleDbSchemaTable

[C#] public DataTable GetOleDbSchemaTable(Guid schema, object[] restrictions);

[C++] public: DataTable\* GetOleDbSchemaTable(Guid schema, Object\* restrictions \_\_gc[]);

[VB] Public Function GetOleDbSchemaTable(ByVal schema As Guid, ByVal restrictions() As Object) As DataTable

[JScript] public function GetOleDbSchemaTable(schema : Guid, restrictions : Object[]) : DataTable;

#### Description

Returns the schema table and associated restriction columns of the specified schema.

**Return Value:** A **System.Data.DataTable** containing a list of schema restrictions.

The schema table is returned as a **System.Data.DataTable** that has the same format as the OLE DB schema rowset specified by the the *schema* parameter. Use the *restrictions* parameter to filter the rows to be returned in the **System.Data.DataTable** (for example, by specifying restrictions for tablename, type, owner, or schema). When you pass values in the array, include empty strings for array elements that do not contain values. If you pass an empty array to *restrictions* , all rows (one for each table) are returned in default order. Values in

the array correspond to the order of the columns in the source table and **System.Data.DataTable** . One of the **System.Data.OleDb.OleDbSchemaGuid** values that specifies the schema table to return. An array of objects that are filter values, each of which corresponds to a **System.Data.DataColumn** in the resulting **System.Data.DataTable** .

## Open

[C#] public void Open();

[C++] public: \_\_sealed void Open();

[VB] NotOverridable Public Sub Open()

[JScript] public function Open();

## Description

Opens a database connection with the property settings specified by the **System.Data.OleDb.OleDbConnection.ConnectionString** .

The **System.Data.OleDb.OleDbConnection** draws an open connection from the connection pool if one is available. Otherwise, it establishes a new connection to the data source.

## ReleaseObjectPool

[C#] public static void ReleaseObjectPool();

[C++] public: static void ReleaseObjectPool();

[VB] Public Shared Sub ReleaseObjectPool()

[JScript] public static function ReleaseObjectPool();

1  
2 *Description*

3 Indicates that the **System.Data.OleDb.OleDbConnection** object pooling  
4 can be cleared when the last underlying OLE DB Provider is released.

5 The object pool is cached whenever one of the underlying OLE DB  
6 providers is created. This method should be called when the user is done using any  
7 **System.Data.OleDb.OleDbConnection** objects.

8 **IDbConnection.BeginTransaction**

9  
10 [C#] IDbTransaction IDbConnection.BeginTransaction();

11 [C++] IDbTransaction\* IDbConnection::BeginTransaction();

12 [VB] Function BeginTransaction() As IDbTransaction Implements

13 IDbConnection.BeginTransaction

14 [JScript] function IDbConnection.BeginTransaction() : IDbTransaction;

15 **IDbConnection.BeginTransaction**

16  
17 [C#] IDbTransaction IDbConnection.BeginTransaction(IsolationLevel  
18 isolationLevel);

19 [C++] IDbTransaction\* IDbConnection::BeginTransaction(IsolationLevel  
20 isolationLevel);

21 [VB] Function BeginTransaction(ByVal isolationLevel As IsolationLevel) As

22 IDbTransaction Implements IDbConnection.BeginTransaction

23 [JScript] function IDbConnection.BeginTransaction(isolationLevel :  
24 IsolationLevel) : IDbTransaction;

25 **IDbConnection.CreateCommand**

```

1
2 [C#] IDbCommand IDbConnection.CreateCommand();
3 [C++] IDbCommand* IDbConnection::CreateCommand();
4 [VB] Function CreateCommand() As IDbCommand Implements
5 IDbConnection.CreateCommand
6 [JScript] function IDbConnection.CreateCommand() : IDbCommand;
7     ICloneable.Clone
8
9 [C#] object ICloneable.Clone();
10 [C++] Object* ICloneable::Clone();
11 [VB] Function Clone() As Object Implements ICloneable.Clone
12 [JScript] function ICloneable.Clone() : Object;
13     OleDbDataAdapter class (System.Data.OleDb)
14     ToString
15
16

```

### *Description*

Represents a set of data commands and a database connection which are used to fill the **System.Data.DataSet** and update the data source.

The **System.Data.OleDb.OleDbDataAdapter** serves as a bridge between a **System.Data.DataSet** and data source for retrieving and saving data. The **System.Data.OleDb.OleDbDataAdapter** provides this bridge by using **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** to load data from the data source into the **System.Data.DataSet** , and using

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to send changes made in the **System.Data.DataSet** back to the data source.

OleDbDataAdapter

*Example Syntax:*

ToString

[C#] public OleDbDataAdapter();

[C++] public: OleDbDataAdapter();

[VB] Public Sub New()

[JScript] public function OleDbDataAdapter(); Initializes a new instance of the **System.Data.OleDb.OleDbDataAdapter** class.

### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbDataAdapter** class.

When you create an instance of **System.Data.OleDb.OleDbDataAdapter** , the following read/write properties are set to the following initial values.

OleDbDataAdapter

*Example Syntax:*

ToString

[C#] public OleDbDataAdapter(OleDbCommand selectCommand);

[C++] public: OleDbDataAdapter(OleDbCommand\* selectCommand);

[VB] Public Sub New(ByVal selectCommand As OleDbCommand)

[JScript] public function OleDbDataAdapter(selectCommand : OleDbCommand);

1  
2 *Description*

3        Initializes a new instance of the **System.Data.OleDb.OleDbDataAdapter**  
4 class with the specified SQL SELECT statement.

5        When you create an instance of **System.Data.OleDb.OleDbDataAdapter** ,  
6 the following read/write properties are set to the following initial values. An  
7 **System.Data.OleDb.OleDbCommand** that is a SQL SELECT statement.

8        OleDbDataAdapter

9        *Example Syntax:*

10       ToString

11  
12 [C#] public OleDbDataAdapter(string selectCommandText, OleDbConnection  
13 selectConnection);

14 [C++] public: OleDbDataAdapter(String\* selectCommandText,  
15 OleDbConnection\* selectConnection);

16 [VB] Public Sub New(ByVal selectCommandText As String, ByVal  
17 selectConnection As OleDbConnection)

18 [JScript] public function OleDbDataAdapter(selectCommandText : String,  
19 selectConnection : OleDbConnection);

20  
21 *Description*

22        Initializes a new instance of the **System.Data.OleDb.OleDbDataAdapter**  
23 class with a **System.Data.OleDb.OleDbDataAdapter.SelectCommand** .

24        This implementation of the **System.Data.OleDb.OleDbDataAdapter**  
25 opens and closes a **System.Data.OleDb.OleDbConnection** if it is not already

open. This can be useful in a an application that must call the **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method for two or more **System.Data.OleDb.OleDbDataAdapter** objects. If the **System.Data.OleDb.OleDbConnection** is already open, you must explicitly call **System.Data.OleDb.OleDbConnection.Close** or **System.Data.OleDb.OleDbConnection.Dispose(System.Boolean)** to close it. The **System.Data.OleDb.OleDbDataAdapter.SelectCommand** . An **System.Data.OleDb.OleDbConnection** that represents the connection.

**OleDbDataAdapter**

*Example Syntax:*

**ToString**

```
[C#] public OleDbDataAdapter(string selectCommandText, string
selectConnectionString);
[C++] public: OleDbDataAdapter(String* selectCommandText, String*
selectConnectionString);
[VB] Public Sub New(ByVal selectCommandText As String, ByVal
selectConnectionString As String)
[JScript] public function OleDbDataAdapter(selectCommandText : String,
selectConnectionString : String);
```

### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbDataAdapter** class with a **System.Data.OleDb.OleDbDataAdapter.SelectCommand** .

When you create an instance of **System.Data.OleDb.OleDbDataAdapter** , the following read/write properties are set to the following initial values. The **System.Data.OleDb.OleDbDataAdapter.SelectCommand** . The connection string.

**AcceptChangesDuringFill**

**Container**

**DeleteCommand**

**ToString**

#### *Description*

Gets or sets an SQL statement or stored procedure for deleting records from the data set.

**During**

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** , if this property is not set and primary key information is present in the

**System.Data.DataSet** , the

**System.Data.OleDb.OleDbDataAdapter.DeleteCommand** can be generated automatically if you set the

**System.Data.OleDb.OleDbDataAdapter.SelectCommand** property and use the **System.Data.OleDb.OleDbCommandBuilder** . Then, any additional commands that you do not set are generated by the

**System.Data.OleDb.OleDbCommandBuilder** . This generation logic requires key column information to be present in the **System.Data.DataSet** . For more information see .

1 Gets or sets the name of the source column mapped to the  
2 **System.Data.DataSet** and used for loading or returning the  
3 **System.Data.OleDb.OleDbParameter.Value** .

4 The link between the value of the **System.Data.OleDb.OleDbParameter**  
5 and the **System.Data.DataTable** may be bidirectional depending on the value of  
6 the **System.Data.OleDb.OleDbParameter.Direction** property.

7 SourceVersion

8 ToString

9  
10 [C#] public DataRowVersion SourceVersion {get; set;}

11 [C++] public: \_\_property DataRowVersion get\_SourceVersion();public:

12 \_\_property void set\_SourceVersion(DataRowVersion);

13 [VB] Public Property SourceVersion As DataRowVersion

14 [JScript] public function get SourceVersion() : DataRowVersion;public function

15 set SourceVersion(DataRowVersion);

16  
17 *Description*

18 Gets or sets the **System.Data.DataRowVersion** to use when loading  
19 **System.Data.OleDb.OleDbParameter.Value** .

20 Used by **System.Data.OleDb.OleDbDataAdapter.UpdateCommand**  
21 during an

22 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**

23 operation to determine whether the parameter value is set to **Current** or **Original** .

24 This allows primary keys to be updated. This property is ignored by

25 **System.Data.OleDb.OleDbDataAdapter.InsertCommand** and

**System.Data.OleDb.OleDbDataAdapter.DeleteCommand** . This property is set to the version of the **System.Data.DataRow** used by the **System.Data.DataRow.Item(System.Int32)** property, or the **System.Data.DataRow.GetChildRows(System.String)** method of the **System.Data.DataRow** object.

Value

ToString

[C#] public object Value {get; set;}

[C++] public: \_\_property Object\* get\_Value();public: \_\_property void set\_Value(Object\*);

[VB] Public Property Value As Object

[JScript] public function get Value() : Object;public function set Value(Object);

### *Description*

Gets or sets the value of the parameter.

For input parameters, the value is bound to the

**System.Data.OleDb.OleDbCommand** that is sent to the server. For output and return value parameters, the value is set on completion of the **System.Data.OleDb.OleDbCommand** and after the **System.Data.OleDb.OleDbDataReader** is closed.

ICloneable.Clone

[C#] object ICloneable.Clone();

[C++] Object\* ICloneable::Clone();

- 1 DesignMode
- 2 Events
- 3 InsertCommand
- 4 ToString

5

6

7 *Description*

8 Gets or sets an SQL statement or stored procedure used to insert new

9 records into the data source.

10 During

11 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** , if this

12 property is not set and primary key information is present in the

13 **System.Data.DataSet** , the

14 **System.Data.OleDb.OleDbDataAdapter.InsertCommand** can be generated

15 automatically if you set the

16 **System.Data.OleDb.OleDbDataAdapter.SelectCommand** property and use the

17 **System.Data.OleDb.OleDbCommandBuilder** . Then, any additional commands

18 that you do not set are generated by the

19 **System.Data.OleDb.OleDbCommandBuilder** . This generation logic requires

20 key column information to be present in the **System.Data.DataSet** . For more

21 information see .

- 22 MissingMappingAction
- 23 MissingSchemaAction
- 24 SelectCommand
- 25 ToString

### Description

Gets or sets an SQL statement or stored procedure used to select records in the data source.

When **System.Data.OleDb.OleDbDataAdapter.SelectCommand** is assigned to a previously created **System.Data.OleDb.OleDbCommand**, the **System.Data.OleDb.OleDbCommand** is not cloned. The **System.Data.OleDb.OleDbDataAdapter.SelectCommand** maintains a reference to the previously created **System.Data.OleDb.OleDbCommand** object.

Site

TableMappings

UpdateCommand

ToString

### Description

Gets or sets an SQL statement or stored procedure used to update records in the data source.

During

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**, if this property is not set and primary key information is present in the **System.Data.DataSet**, the **System.Data.OleDb.OleDbDataAdapter.UpdateCommand** can be generated automatically if you set the

**System.Data.OleDb.OleDbDataAdapter.SelectCommand** property and use the **System.Data.OleDb.OleDbCommandBuilder** . Then, any additional commands that you do not set are generated by the **System.Data.OleDb.OleDbCommandBuilder** . This generation logic requires key column information to be present in the **System.Data.DataSet** . For more information see .

ToString

#### *Description*

Occurs during

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** after a command is executed against the data source. The attempt to update is made, so the event fires.

When using

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** , there are two events that occur per data row updated. The order of execution is as follows: The values in the **System.Data.DataRow** are moved to the parameter values.

ToString

[C#] public event OleDbRowUpdatingEventHandler RowUpdating;

[C++] public: \_\_event OleDbRowUpdatingEventHandler\* RowUpdating;

[VB] Public Event RowUpdating As OleDbRowUpdatingEventHandler

## Description

Occurs during

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** before a command is executed against the data source. The attempt to update is made, so the event fires.

When using

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**, there are two events that occur per data row updated. The order of execution is as follows: The values in the **System.Data.DataRow** are moved to the parameter values.

CreateRowUpdatedEvent

[C#] protected override RowUpdatedEventArgs

CreateRowUpdatedEvent(DataRow dataRow, IDbCommand command, StatementType statementType, DataTableMapping tableMapping);

[C++] protected: RowUpdatedEventArgs\* CreateRowUpdatedEvent(DataRow\* dataRow, IDbCommand\* command, StatementType statementType, DataTableMapping\* tableMapping);

[VB] Overrides Protected Function CreateRowUpdatedEvent(ByVal dataRow As DataRow, ByVal command As IDbCommand, ByVal statementType As StatementType, ByVal tableMapping As DataTableMapping) As RowUpdatedEventArgs

[JScript] protected override function CreateRowUpdatedEvent(dataRow : DataRow, command : IDbCommand, statementType : StatementType,

tableMapping : DataTableMapping) : RowUpdatedEventArgs;

*Description*

CreateRowUpdatingEvent

[C#] protected override RowUpdatingEventArgs

CreateRowUpdatingEvent(DataRow dataRow, IDbCommand command,

StatementType statementType, DataTableMapping tableMapping);

[C++] protected: RowUpdatingEventArgs\* CreateRowUpdatingEvent(DataRow\*

dataRow, IDbCommand\* command, StatementType statementType,

DataTableMapping\* tableMapping);

[VB] Overrides Protected Function CreateRowUpdatingEvent(ByVal dataRow As

DataRow, ByVal command As IDbCommand, ByVal statementType As

StatementType, ByVal tableMapping As DataTableMapping) As

RowUpdatingEventArgs

[JScript] protected override function CreateRowUpdatingEvent(dataRow :

DataRow, command : IDbCommand, statementType : StatementType,

tableMapping : DataTableMapping) : RowUpdatingEventArgs;

*Description*

Dispose

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean); Releases the resources used by the **System.Data.OleDb.OleDbDataAdapter** .

*Description*

Releases the unmanaged resources used by the **System.Data.OleDb.OleDbDataAdapter** and optionally releases the managed resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

**Fill**

[C#] public int Fill(DataTable dataTable, object adodb);

[C++] public: int Fill(DataTable\* dataTable, Object\* adodb);

[VB] Public Function Fill(ByVal dataTable As DataTable, ByVal adodb As

Object) As Integer

[JScript] public function Fill(dataTable : DataTable, adodb : Object) : int; Adds or refreshes rows in the **System.Data.DataSet** to match those in an ADO **Recordset** or **Record** object.

*Description*

Adds or refreshes rows in a **System.Data.DataTable** to match those in an ADO **Recordset** or **Record** object using the specified **System.Data.DataTable** and ADO objects.

*Return Value:* The number of rows successfully refreshed to the

**System.Data.DataTable** . This does not include rows affected by statements that do not return rows.

The link between ActiveX Data Objects (ADO) and ADO.NET is a one-way operation in that you can copy data from ADO to the **System.Data.DataSet** , but any updates to the data must be handled by ADO.NET. For more information, see . A **System.Data.DataTable** to fill with records and, if necessary, schema. An ADO **Recordset** or **Record** object.

Fill

```
[C#] public int Fill(DataSet dataSet, object adodb, string srcTable);
[C++] public: int Fill(DataSet* dataSet, Object* adodb, String* srcTable);
[VB] Public Function Fill(ByVal dataSet As DataSet, ByVal adodb As Object,
ByVal srcTable As String) As Integer
[JScript] public function Fill(dataSet : DataSet, adodb : Object, srcTable : String) :
int;
```

### *Description*

Adds or refreshes rows in the **System.Data.DataSet** to match those in an ADO **Recordset** or **Record** object using the specified **System.Data.DataSet** , ADO object, and source table name.

*Return Value:* The number of rows successfully added to or refreshed in the **System.Data.DataSet** . This does not include rows affected by statements that do not return rows.

The link between ActiveX Data Objects (ADO) and ADO.NET is a one-way operation in that you can copy data from ADO to the **System.Data.DataSet** ,

but any updates to the data must be handled by ADO.NET. For more information, see . A **System.Data.DataSet** to fill with records and, if necessary, schema. An **ADO Recordset** or **Record** object. The source table used for the table mappings.

#### OnRowUpdated

[C#] protected override void OnRowUpdated(RowUpdatedEventArgs value);

[C++] protected: void OnRowUpdated(RowUpdatedEventArgs\* value);

[VB] Overrides Protected Sub OnRowUpdated(ByVal value As RowUpdatedEventArgs)

[JScript] protected override function OnRowUpdated(value : RowUpdatedEventArgs);

#### Description

Raises the **System.Data.OleDb.OleDbDataAdapter.OnRowUpdated(System.Data.Common.RowUpdatedEventArgs)** event using a **System.Data.Common.RowUpdatedEventArgs** object.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Data.Common.RowUpdatedEventArgs** that contains the event data.

#### OnRowUpdating

[C#] protected override void OnRowUpdating(RowUpdatingEventArgs value);

[C++] protected: void OnRowUpdating(RowUpdatingEventArgs\* value);

[VB] Overrides Protected Sub OnRowUpdating(ByVal value As

RowUpdatingEventArgs)

[JScript] protected override function OnRowUpdating(value :  
RowUpdatingEventArgs);

### *Description*

Raises the

**System.Data.OleDb.OleDbDataAdapter.OnRowUpdating(System.Data.Common.RowUpdatingEventArgs)** event using a  
**System.Data.Common.RowUpdatingEventArgs** object.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Data.Common.RowUpdatingEventArgs** that contains the event data.

ICloneable.Clone

[C#] object ICloneable.Clone();

[C++] Object\* ICloneable::Clone();

[VB] Function Clone() As Object Implements ICloneable.Clone

[JScript] function ICloneable.Clone() : Object;

OleDbDataReader class (System.Data.OleDb)

Update

### *Description*

Provides a way of reading a forward-only stream of data rows from a data source. This class cannot be inherited.

To create an **System.Data.OleDb.OleDbDataReader** , you must call the **System.Data.OleDb.OleDbCommand.ExecuteReader** method of the **System.Data.OleDb.OleDbCommand** object, rather than directly using a constructor.

Depth

Update

[C#] public int Depth {get;}

[C++] public: \_\_property int get\_Depth();

[VB] Public ReadOnly Property Depth As Integer

[JScript] public function get Depth() : int;

#### *Description*

Gets a value indicating the depth of nesting for the current row.

The outermost table has a depth of zero.

FieldCount

Update

[C#] public int FieldCount {get;}

[C++] public: \_\_property int get\_FieldCount();

[VB] Public ReadOnly Property FieldCount As Integer

[JScript] public function get FieldCount() : int;

#### *Description*

Gets the number of columns in the current row.

After executing a query that does not return rows (for example, using the **System.Data.OleDb.OleDbCommand.ExecuteNonQuery** method), **System.Data.OleDb.OleDbDataReader.FieldCount** returns -1.

IsClosed

Update

[C#] public bool IsClosed {get;}

[C++] public: \_\_property bool get\_IsClosed();

[VB] Public ReadOnly Property IsClosed As Boolean

[JScript] public function get IsClosed() : Boolean;

#### *Description*

Indicates whether the data reader is closed.

**System.Data.OleDb.OleDbDataReader.IsClosed** and **System.Data.OleDb.OleDbDataReader.RecordsAffected** are the only properties that you can call after the **System.Data.OleDb.OleDbDataReader** is closed.

Item

Update

[C#] public object this[string name] {get;}

[C++] public: \_\_property Object\* get\_Item(String\* name);

[VB] Public Default ReadOnly Property Item(ByVal name As String) As Object

[JScript] returnValue = OleDbDataReaderObject.Item(name);

#### *Description*

Gets the value of the specified column in its native format given the column name. The column name.

Item

Update

[C#] public object this[int index] {get;}

[C++] public: \_\_property Object\* get\_Item(int index);

[VB] Public Default ReadOnly Property Item(ByVal index As Integer) As Object

[JScript] returnValue = OleDbDataReaderObject.Item(index); Gets the value of a column in its native format.

#### *Description*

Gets the value of the specified column in its native format given the column ordinal. The column ordinal.

RecordsAffected

Update

[C#] public int RecordsAffected {get;}

[C++] public: \_\_property int get\_RecordsAffected();

[VB] Public ReadOnly Property RecordsAffected As Integer

[JScript] public function get RecordsAffected() : int;

#### *Description*

Gets the number of rows changed, inserted, or deleted by execution of the SQL statement.

1       The **System.Data.OleDb.OleDbDataReader.RecordsAffected** property is  
2 not set until all rows are read and you close the  
3 **System.Data.OleDb.OleDbDataReader** .

4       Close

5  
6    [C#] public void Close();  
7    [C++] public: \_\_sealed void Close();  
8    [VB] NotOverridable Public Sub Close()  
9    [JScript] public function Close();

10  
11    *Description*

12       Closes the **System.Data.OleDb.OleDbDataReader** object.

13       You must explicitly call the  
14 **System.Data.OleDb.OleDbDataReader.Close** method when you are through  
15 using the **System.Data.OleDb.OleDbDataReader** to use the associated  
16 **System.Data.OleDb.OleDbConnection** for any other purpose.

17       Finalize

18  
19    [C#] ~OleDbDataReader();  
20    [C++] ~OleDbDataReader();  
21    [VB] Overrides Protected Sub Finalize()  
22    [JScript] protected override function Finalize();

23  
24    *Description*  
25

1 Frees resources before the **System.Data.OleDb.OleDbDataReader** is  
2 reclaimed by the Garbage Collector.

### 3 **GetBoolean**

4  
5 [C#] public bool GetBoolean(int ordinal);

6 [C++] public: \_\_sealed bool GetBoolean(int ordinal);

7 [VB] NotOverridable Public Function GetBoolean(ByVal ordinal As Integer) As  
8 Boolean

9 [JScript] public function GetBoolean(ordinal : int) : Boolean;

### 10 11 *Description*

12 Gets the value of the specified column as a boolean.

13 *Return Value:* The value of the column.

14 No conversions are performed, therefore the data retrieved must already be  
15 a boolean or an exception is generated. The zero-based column ordinal.

### 16 **GetByte**

17  
18 [C#] public byte GetByte(int ordinal);

19 [C++] public: \_\_sealed unsigned char GetByte(int ordinal);

20 [VB] NotOverridable Public Function GetByte(ByVal ordinal As Integer) As Byte

21 [JScript] public function GetByte(ordinal : int) : Byte;

### 22 23 *Description*

24 Gets the value of the specified column as a byte.

25 *Return Value:* The value of the specified column as a byte.

No conversions are performed, therefore the data retrieved must already be a byte. The zero-based column ordinal.

### GetBytes

```
[C#] public long GetBytes(int ordinal, long dataIndex, byte[] buffer, int  
bufferIndex, int length);
```

```
[C++] public: __sealed __int64 GetBytes(int ordinal, __int64 dataIndex, unsigned  
char buffer __gc[], int bufferIndex, int length);
```

```
[VB] NotOverridable Public Function GetBytes(ByVal ordinal As Integer, ByVal  
dataIndex As Long, ByVal buffer() As Byte, ByVal bufferIndex As Integer,  
ByVal length As Integer) As Long
```

```
[JScript] public function GetBytes(ordinal : int, dataIndex : long, buffer : Byte[],  
bufferIndex : int, length : int) : long;
```

### *Description*

Reads a stream of bytes from the offset specified column offset into the buffer as an array starting at the given buffer offset.

*Return Value:* The actual number of bytes read.

The actual number of bytes read can be less than the requested length, if the end of the row is reached. If you pass a buffer that is **null**,

**System.Data.OleDb.OleDbDataReader.GetBytes(System.Int32, System.Int64, System.Byte[], System.Int32, System.Int32)** returns the length of the row in bytes.

The zero-based column ordinal. The index within the field from which to begin the read operation. The buffer into which to read the stream of bytes. The index for *buffer* to begin the read operation. The number of bytes to read.

## GetChar

[C#] public char GetChar(int ordinal);

[C++] public: \_\_sealed \_\_wchar\_t GetChar(int ordinal);

[VB] NotOverridable Public Function GetChar(ByVal ordinal As Integer) As Char

[JScript] public function GetChar(ordinal : int) : Char;

### *Description*

Gets the value of the specified column as a character.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a character. The zero-based column ordinal.

## GetChars

[C#] public long GetChars(int ordinal, long dataIndex, char[] buffer, int bufferSize, int length);

[C++] public: \_\_sealed \_\_int64 GetChars(int ordinal, \_\_int64 dataIndex, \_\_wchar\_t buffer \_\_gc[], int bufferSize, int length);

[VB] NotOverridable Public Function GetChars(ByVal ordinal As Integer, ByVal dataIndex As Long, ByVal buffer() As Char, ByVal bufferSize As Integer, ByVal length As Integer) As Long

[JScript] public function GetChars(ordinal : int, dataIndex : long, buffer : Char[], bufferSize : int, length : int) : long;

### *Description*

1 Reads a stream of characters from the specified column offset into the  
2 buffer as an array starting at the given buffer offset.

3 *Return Value:* The actual number of characters read.

4 The actual number of characters read can be less than the requested length,  
5 if the end of the field is reached. If you pass a buffer that is **null** ,  
6 **System.Data.OleDb.OleDbDataReader.GetChars(System.Int32,System.Int64,**  
7 **System.Char[],System.Int32,System.Int32)** returns the length of the field in  
8 characters. The zero-based column ordinal. The index within the row from which  
9 to begin the read operation. The buffer into which to copy data. The index for  
10 *buffer* to begin the read operation. The number of characters to read.

#### 11 GetData

12  
13 [C#] public OleDbDataReader GetData(int ordinal);

14 [C++] public: OleDbDataReader\* GetData(int ordinal);

15 [VB] Public Function GetData(ByVal ordinal As Integer) As OleDbDataReader

16 [JScript] public function GetData(ordinal : int) : OleDbDataReader;

#### 17 18 Description

19 Not currently supported. The zero-based column ordinal.

#### 20 GetDataTypeName

21  
22 [C#] public string GetDataTypeName(int index);

23 [C++] public: \_\_sealed String\* GetDataTypeName(int index);

24 [VB] NotOverridable Public Function GetDataTypeName(ByVal index As  
25 Integer) As String

1 [JScript] public function GetDataTypeName(index : int) : String;

2

3 *Description*

4 Gets the name of the source data type.

5 *Return Value:* The name of the back-end data type. The zero-based column  
6 ordinal.

7       GetDateTime

8

9 [C#] public DateTime GetDateTime(int ordinal);

10 [C++] public: \_\_sealed DateTime GetDateTime(int ordinal);

11 [VB] NotOverridable Public Function GetDateTime(ByVal ordinal As Integer) As  
12 DateTime

13 [JScript] public function GetDateTime(ordinal : int) : DateTime;

14

15 *Description*

16 Gets the value of the specified column as a **System.DateTime** object.

17 *Return Value:* The value of the specified column.

18 No conversions are performed, therefore the data retrieved must already be  
19 a **System.DateTime** object. The zero-based column ordinal.

20       GetDecimal

21

22 [C#] public decimal GetDecimal(int ordinal);

23 [C++] public: \_\_sealed Decimal GetDecimal(int ordinal);

24 [VB] NotOverridable Public Function GetDecimal(ByVal ordinal As Integer) As  
25 Decimal

1 [JScript] public function GetDecimal(ordinal : int) : Decimal;

2

3 *Description*

4 Gets the value of the specified column as a **System.Decimal** object.

5 *Return Value:* The value of the specified column.

6 No conversions are performed, therefore the data retrieved must already be  
7 a **System.Decimal** object. The zero-based column ordinal.

8 **GetDouble**

9

10 [C#] public double GetDouble(int ordinal);

11 [C++] public: \_\_sealed double GetDouble(int ordinal);

12 [VB] NotOverridable Public Function GetDouble(ByVal ordinal As Integer) As  
13 Double

14 [JScript] public function GetDouble(ordinal : int) : double;

15

16 *Description*

17 Gets the value of the specified column as a double-precision floating point  
18 number.

19 *Return Value:* The value of the specified column.

20 No conversions are performed, therefore the data retrieved must already be  
21 a double-precision floating point number. The zero-based column ordinal.

22 **GetFieldType**

23

24 [C#] public Type GetFieldType(int index);

25 [C++] public: \_\_sealed Type\* GetFieldType(int index);

[VB] NotOverridable Public Function GetFieldType(ByVal index As Integer) As  
Type

[JScript] public function GetFieldType(index : int) : Type;

*Description*

Gets the **System.Type** that is the data type of the object.

*Return Value:* The **System.Type** that is the data type of the object. The zero-based column ordinal.

GetFloat

[C#] public float GetFloat(int ordinal);

[C++] public: \_\_sealed float GetFloat(int ordinal);

[VB] NotOverridable Public Function GetFloat(ByVal ordinal As Integer) As  
Single

[JScript] public function GetFloat(ordinal : int) : float;

*Description*

Gets the value of the specified column as a single-precision floating point number.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a single-precision floating point number. The zero-based column ordinal.

GetGuid

[C#] public Guid GetGuid(int ordinal);

[C++] public: \_\_sealed Guid GetGuid(int ordinal);

[VB] NotOverridable Public Function GetGuid(ByVal ordinal As Integer) As Guid

[JScript] public function GetGuid(ordinal : int) : Guid;

### *Description*

Gets the value of the specified column as a globally-unique identifier (GUID).

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a globally-unique identifier. The zero-based column ordinal.

### *GetInt16*

[C#] public short GetInt16(int ordinal);

[C++] public: \_\_sealed short GetInt16(int ordinal);

[VB] NotOverridable Public Function GetInt16(ByVal ordinal As Integer) As Short

[JScript] public function GetInt16(ordinal : int) : Int16;

### *Description*

Gets the value of the specified column as a 16-bit signed integer.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a 16-bit signed integer. The zero-based column ordinal.

### *GetInt32*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public int GetInt32(int ordinal);
[C++] public: __sealed int GetInt32(int ordinal);
[VB] NotOverridable Public Function GetInt32(ByVal ordinal As Integer) As Integer
[JScript] public function GetInt32(ordinal : int) : int;
```

*Description*

Gets the value of the specified column as a 32-bit signed integer.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a 32-bit signed integer. The zero-based column ordinal.

**GetInt64**

```
[C#] public long GetInt64(int ordinal);
[C++] public: __sealed __int64 GetInt64(int ordinal);
[VB] NotOverridable Public Function GetInt64(ByVal ordinal As Integer) As Long
[JScript] public function GetInt64(ordinal : int) : long;
```

*Description*

Gets the value of the specified column as a 64-bit signed integer.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a 64-bit signed integer. The zero-based column ordinal.

```

1      GetName
2
3      [C#] public string GetName(int index);
4      [C++] public: __sealed String* GetName(int index);
5      [VB] NotOverridable Public Function GetName(ByVal index As Integer) As
6      String
7      [JScript] public function GetName(index : int) : String;
8
9      Description
10         Gets the name of the specified column.
11      Return Value: The name of the specified column. The zero-based column ordinal.
12
13      GetOrdinal
14
15      [C#] public int GetOrdinal(string name);
16      [C++] public: __sealed int GetOrdinal(String* name);
17      [VB] NotOverridable Public Function GetOrdinal(ByVal name As String) As
18      Integer
19      [JScript] public function GetOrdinal(name : String) : int;
20
21      Description
22         Gets the column ordinal, given the name of the column.
23      Return Value: The zero-based column ordinal. The name of the column.
24
25      GetSchemaTable
26
27      [C#] public DataTable GetSchemaTable();

```

[C++] public: \_\_sealed DataTable\* GetSchemaTable();

[VB] NotOverridable Public Function GetSchemaTable() As DataTable

[JScript] public function GetSchemaTable() : DataTable;

### *Description*

Returns a **System.Data.DataTable** that describes the column metadata of the **System.Data.OleDb.OleDbDataReader**.

*Return Value:* A **System.Data.DataTable** that describes the column metadata.

The **System.Data.OleDb.OleDbDataReader.GetSchemaTable** method maps to the OLE DB **IColumnsRowset::GetColumnsRowset** method, and returns metadata about each column in the following order: **DataReader Column OLE DB Column ID Description ColumnName DBCOLUMN\_NAME** The name of the column; this might not be unique. If this cannot be determined, a null value is returned. This name always reflects the most recent renaming of the column in the current view or command text.

### **GetString**

[C#] public string GetString(int ordinal);

[C++] public: \_\_sealed String\* GetString(int ordinal);

[VB] NotOverridable Public Function GetString(ByVal ordinal As Integer) As String

[JScript] public function GetString(ordinal : int) : String;

### *Description*

Gets the value of the specified column as a string.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a string. The zero-based column ordinal.

GetTimeSpan

[C#] public TimeSpan GetTimeSpan(int ordinal);

[C++] public: TimeSpan GetTimeSpan(int ordinal);

[VB] Public Function GetTimeSpan(ByVal ordinal As Integer) As TimeSpan

[JScript] public function GetTimeSpan(ordinal : int) : TimeSpan;

*Description*

Gets the value of the specified column as a **System.TimeSpan** object.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a **System.TimeSpan** object. The zero-based column ordinal.

GetValue

[C#] public object GetValue(int ordinal);

[C++] public: \_\_sealed Object\* GetValue(int ordinal);

[VB] NotOverridable Public Function GetValue(ByVal ordinal As Integer) As

Object

[JScript] public function GetValue(ordinal : int) : Object; Gets the value of the specified column in its native format.

## Description

Gets the value of the column at the specified ordinal in its native format.

**Return Value:** The value to return. The zero-based column ordinal.

GetValues

[C#] public int GetValues(object[] values);

[C++] public: \_\_sealed int GetValues(Object\* values \_\_gc[]);

[VB] NotOverridable Public Function GetValues(ByVal values() As Object) As

Integer

[JScript] public function GetValues(values : Object[]) : int;

## Description

Gets all the attribute columns in the current row.

**Return Value:** The number of instances of **System.Object** in the array.

For most applications, the

**System.Data.OleDb.OleDbDataReader.GetValues(System.Object[])** method provides an efficient means for retrieving all columns, rather than retrieving each column individually. An array of **System.Object** into which to copy the attribute columns.

IsDBNull

[C#] public bool IsDBNull(int ordinal);

[C++] public: \_\_sealed bool IsDBNull(int ordinal);

[VB] NotOverridable Public Function IsDBNull(ByVal ordinal As Integer) As

Boolean

[JScript] public function IsDBNull(ordinal : int) : Boolean;

*Description*

Gets a value indicating whether the column contains non-existent or missing values.

**Return Value:** **true** if the specified column value is equivalent to **System.DBNull** ; otherwise, **false** . The zero-based column ordinal.

NextResult

[C#] public bool NextResult();

[C++] public: \_\_sealed bool NextResult();

[VB] NotOverridable Public Function NextResult() As Boolean

[JScript] public function NextResult() : Boolean;

*Description*

Advances the data reader to the next result, when reading the results of batch SQL statements.

**Return Value:** **true** if there are more rows; otherwise, **false** .

Used to process multiple results, which can be generated by executing batch SQL statements.

Read

[C#] public bool Read();

[C++] public: \_\_sealed bool Read();

[VB] NotOverridable Public Function Read() As Boolean

[JScript] public function Read() : Boolean;

#### *Description*

Advances the **System.Data.OleDb.OleDbDataReader** to the next record.

*Return Value:* **true** if there are more rows; otherwise, **false** .

The default position of the **System.Data.OleDb.OleDbDataReader** is prior to the first record. Therefore, you must call

**System.Data.OleDb.OleDbDataReader.Read** to begin accessing any data.

**IEnumerator.GetEnumerator**

[C#] IEnumerator IEnumerable.GetEnumerator();

[C++] IEnumerator\* IEnumerable::GetEnumerator();

[VB] Function GetEnumerator() As IEnumerator Implements

**IEnumerator.GetEnumerator**

[JScript] function IEnumerable.GetEnumerator() : IEnumerator;

**IDataRecord.GetData**

[C#] IDataReader IDataRecord.GetData(int ordinal);

[C++] IDataReader\* IDataRecord::GetData(int ordinal);

[VB] Function GetData(ByVal ordinal As Integer) As IDataReader Implements

**IDataRecord.GetData**

[JScript] function IDataRecord.GetData(ordinal : int) : IDataReader;

**IDisposable.Dispose**

1  
2 [C#] void IDisposable.Dispose();  
3 [C++] void IDisposable::Dispose();  
4 [VB] Sub Dispose() Implements IDisposable.Dispose  
5 [JScript] function IDisposable.Dispose();

6       OleDbError class (System.Data.OleDb)  
7       ToString

8  
9  
10 *Description*

11       Collects information relevant to a warning or error returned by the data  
12 source. This class cannot be inherited.

13       This class is created by the OleDb data adapter when an error occurs. An  
14 instance of **System.Data.OleDb.OleDbError** is created and managed by the  
15 **System.Data.OleDb.OleDbErrorCollection** class, which in turn is created by the  
16 **System.Data.OleDb.OleDbException** class.

17       Message  
18       ToString

19  
20 [C#] public string Message {get;}  
21 [C++] public: \_\_property String\* get\_Message();  
22 [VB] Public ReadOnly Property Message As String  
23 [JScript] public function get Message() : String;

24  
25 *Description*

1 Gets a short description of the error.

2 NativeError

3 ToString

4

5 [C#] public int NativeError {get;}

6 [C++] public: \_\_property int get\_NativeError();

7 [VB] Public ReadOnly Property NativeError As Integer

8 [JScript] public function get NativeError() : int;

9

10 *Description*

11 Gets the database-specific error information.

12 Source

13 ToString

14

15 [C#] public string Source {get;}

16 [C++] public: \_\_property String\* get\_Source();

17 [VB] Public ReadOnly Property Source As String

18 [JScript] public function get Source() : String;

19

20 *Description*

21 Gets the name of the provider that generated the error.

22 SQLState

23 ToString

24

25 [C#] public string SQLState {get;}

1 [C++] public: \_\_property String\* get\_SQLState();

2 [VB] Public ReadOnly Property SQLState As String

3 [JScript] public function get SQLState() : String;

4  
5 *Description*

6 Gets the five-character error code following the ANSI SQL standard for the  
7 database.

8 ToString

9  
10 [C#] public override string ToString();

11 [C++] public: String\* ToString();

12 [VB] Overrides Public Function ToString() As String

13 [JScript] public override function ToString() : String;

14  
15 *Description*

16 Gets the complete text of the error message.

17 *Return Value:* The complete text of the error.

18 The string is in the form "OleDbError:", followed by the

19 **System.Data.OleDb.OleDbError.Message** , and the stack trace. For example:

20 OleDbError:UserId or Password not valid. The following example displays each

21 **System.Data.OleDb.OleDbError** within the

22 **System.Data.OleDb.OleDbErrorCollection** collection.

23 OleDbErrorCollection class (System.Data.OleDb)

24 ToString

### Description

Collects all errors generated by the adapter. This class cannot be inherited.

This class is created by **System.Data.OleDb.OleDbException** to collect instances of the **System.Data.OleDb.OleDbError** class.

Count

ToString

[C#] public int Count {get;}

[C++] public: \_\_property int get\_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

### Description

Gets the number of errors in the collection.

Item

ToString

[C#] public OleDbError this[int index] {get;}

[C++] public: \_\_property OleDbError\* get\_Item(int index);

[VB] Public Default ReadOnly Property Item(ByVal index As Integer) As

OleDbError

[JScript] returnValue = OleDbErrorCollectionObject.Item(index);

## Description

Gets the error at the specified index.

The following example displays each **System.Data.OleDb.OleDbError** within the **System.Data.OleDb.OleDbErrorCollection** collection. The zero-based index of the error to retrieve.

## CopyTo

```
[C#] public void CopyTo(Array array, int index);
```

```
[C++] public: __sealed void CopyTo(Array* array, int index);
```

```
[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)
```

```
[JScript] public function CopyTo(array : Array, index : int);
```

## Description

Copies the elements of the **System.Data.OleDb.OleDbErrorCollection** into an **System.Array** , starting at the given index within the **System.Array** . The **System.Array** into which to copy the elements. The starting index of the *array* .

## GetEnumerator

```
[C#] public IEnumerator GetEnumerator();
```

```
[C++] public: __sealed IEnumerator* GetEnumerator();
```

```
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
```

```
[JScript] public function GetEnumerator() : IEnumerator;
```

1  
2 *Description*

3 OleDbException class (System.Data.OleDb)

4 ToString

5  
6  
7 *Description*

8 The exception that is thrown when a warning or error is returned by an  
9 OLE DB data source. This class cannot be inherited.

10 This class is created whenever the OleDb adapter encounters a situation that  
11 it cannot handle. It always contains at least one instance of

12 **System.Data.OleDb.OleDbError** .

13 ErrorCode

14 ToString

15  
16 [C#] public override int ErrorCode {get;}

17 [C++] public: \_\_property virtual int get\_ErrorMessage();

18 [VB] Overrides Public ReadOnly Property ErrorCode As Integer

19 [JScript] public function get ErrorCode() : int;

20  
21 *Description*

22 Errors

23 ToString

24  
25 [C#] public OleDbErrorCollection Errors {get;}

```

1 [C++] public: __property OleDbErrorCollection* get_Errors();
2 [VB] Public ReadOnly Property Errors As OleDbErrorCollection
3 [JScript] public function get Errors() : OleDbErrorCollection;
4

```

#### 5 *Description*

6 Gets a collection of one or more **System.Data.OleDb.OleDbError** objects  
7 that give detailed information about exceptions generated by the OLE DB .NET  
8 Data Provider.

9 The **System.Data.OleDb.OleDbErrorCollection** class always contains at  
10 least one instance of the **System.Data.OleDb.OleDbError** class.

11 HelpLink

12 HResult

13 InnerException

14 Message

15 ToString

#### 18 *Description*

19 Gets the text describing the error.

20 This is a wrapper for the **System.Data.OleDb.OleDbError.Message**  
21 property of the first **System.Data.OleDb.OleDbError** in the  
22 **System.Data.OleDb.OleDbException.Errors** collection property.

23 Source

24 ToString

25

1  
2 [C#] public override string Source {get;}

3 [C++] public: \_\_property virtual String\* get\_Source();

4 [VB] Overrides Public ReadOnly Property Source As String

5 [JScript] public function get Source() : String;

6  
7 *Description*

8 Gets the name of the provider that generated the error.

9 This is a wrapper for the **System.Data.OleDb.OleDbError.Source**  
10 property of the first **System.Data.OleDb.OleDbError** in the  
11 **System.Data.OleDb.OleDbException.Errors** collection.

12 StackTrace

13 TargetSite

14 **ISerializable.GetObjectData**

15  
16 [C#] void **ISerializable.GetObjectData**(SerializationInfo si, StreamingContext  
17 context);

18 [C++] void **ISerializable::GetObjectData**(SerializationInfo\* si, StreamingContext  
19 context);

20 [VB] Sub **GetObjectData**(ByVal si As SerializationInfo, ByVal context As  
21 StreamingContext) Implements **ISerializable.GetObjectData**

22 [JScript] function **ISerializable.GetObjectData**(si : SerializationInfo, context :  
23 StreamingContext);

24 **OleDbInfoMessageEventArgs** class (System.Data.OleDb)

25 **ToString**

1  
2  
3 *Description*

4 Provides data for the  
5 **System.Data.OleDb.OleDbConnection.InfoMessage** event. This class cannot be  
6 inherited.

7 The **System.Data.OleDb.OleDbConnection.InfoMessage** event contains  
8 an **System.Data.OleDb.OleDbErrorCollection** collection with warnings sent  
9 from the data source.

10 ErrorCode

11 ToString

12  
13 [C#] public int ErrorCode {get;}

14 [C++] public: \_\_property int get\_ErrorCode();

15 [VB] Public ReadOnly Property ErrorCode As Integer

16 [JScript] public function get ErrorCode() : int;

17  
18 *Description*

19 Gets the HRESULT following the ANSI SQL standard for the database.

20 This is a wrapper for the **System.Data.OleDb.OleDbError.SQLState**  
21 property of the first **System.Data.OleDb.OleDbError** in the  
22 **System.Data.OleDb.OleDbInfoMessageEventArgs.Errors** collection.

23 Errors

24 ToString

[illegible]

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

1 [C++] public: \_\_property String\* get\_Source();  
2 [VB] Public ReadOnly Property Source As String  
3 [JScript] public function get Source() : String;  
4

5 *Description*

6 Gets the name of the object that generated the error.

7 This is a wrapper for the **System.Data.OleDb.OleDbError.Source**  
8 property of the first **System.Data.OleDb.OleDbError** in the  
9 **System.Data.OleDb.OleDbInfoMessageEventArgs.Errors** collection.

10 ToString

11  
12 [C#] public override string ToString();  
13 [C++] public: String\* ToString();  
14 [VB] Overrides Public Function ToString() As String  
15 [JScript] public override function ToString() : String;  
16

17 *Description*

18 Retrieves a string representation of the  
19 **System.Data.OleDb.OleDbConnection.InfoMessage** event.

20 *Return Value:* A string representing the

21 **System.Data.OleDb.OleDbConnection.InfoMessage** event.

22 OleDbInfoMessageEventHandler delegate (System.Data.OleDb)

23 ToString  
24  
25

1  
2  
3 *Description*

4 Represents the method that will handle the  
5 **System.Data.OleDb.OleDbConnection.InfoMessage** event of an  
6 **System.Data.OleDb.OleDbConnection** . The source of the event. An  
7 **System.Data.OleDb.OleDbInfoMessageEventArgs** object that contains the  
8 event data.

9 When you create an **System.Data.OleDb.OleDbInfoMessageEventArgs**  
10 delegate, you identify the method that will handle the event. To associate the event  
11 with your event handler, add an instance of the delegate to the event. The event  
12 handler is called whenever the event occurs, unless you remove the delegate. For  
13 more information about event handler delegates, see .

14 OleDbLiteral enumeration (System.Data.OleDb)

15 ToString

16  
17  
18 *Description*

19 Returns information about literals used in text commands, data values, and  
20 database objects.

21 The **System.Data.OleDb.OleDbLiteral** enumeration returns the following  
22 categories of literal information.

23 ToString

24  
25 [C#] public const OleDbLiteral Binary\_Literal;

1 [C++] public: const OleDbLiteral Binary\_Literal;  
 2 [VB] Public Const Binary\_Literal As OleDbLiteral  
 3 [JScript] public var Binary\_Literal : OleDbLiteral;

4  
 5 *Description*

6 A binary literal in a text command.

7 ToString

8  
 9 [C#] public const OleDbLiteral Catalog\_Name;  
 10 [C++] public: const OleDbLiteral Catalog\_Name;  
 11 [VB] Public Const Catalog\_Name As OleDbLiteral  
 12 [JScript] public var Catalog\_Name : OleDbLiteral;

13  
 14 *Description*

15 A catalog name in a text command.

16 ToString

17  
 18 [C#] public const OleDbLiteral Catalog\_Separator;  
 19 [C++] public: const OleDbLiteral Catalog\_Separator;  
 20 [VB] Public Const Catalog\_Separator As OleDbLiteral  
 21 [JScript] public var Catalog\_Separator : OleDbLiteral;

22  
 23 *Description*

24 The character that separates the catalog name from the rest of the identifier  
 25 in a text command.

## ToString

```
[C#] public const OleDbLiteral Char_Literal;
[C++] public: const OleDbLiteral Char_Literal;
[VB] Public Const Char_Literal As OleDbLiteral
[JScript] public var Char_Literal : OleDbLiteral;
```

### *Description*

A character literal in a text command.

## ToString

```
[C#] public const OleDbLiteral Column_Alias;
[C++] public: const OleDbLiteral Column_Alias;
[VB] Public Const Column_Alias As OleDbLiteral
[JScript] public var Column_Alias : OleDbLiteral;
```

### *Description*

A column alias in a text command.

## ToString

```
[C#] public const OleDbLiteral Column_Name;
[C++] public: const OleDbLiteral Column_Name;
[VB] Public Const Column_Name As OleDbLiteral
[JScript] public var Column_Name : OleDbLiteral;
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

A column name used in a text command or in a data-definition interface.

ToString

[C#] public const OleDbLiteral Correlation\_Name;  
[C++] public: const OleDbLiteral Correlation\_Name;  
[VB] Public Const Correlation\_Name As OleDbLiteral  
[JScript] public var Correlation\_Name : OleDbLiteral;

*Description*

A correlation name (table alias) in a text command.

ToString

[C#] public const OleDbLiteral Cube\_Name;  
[C++] public: const OleDbLiteral Cube\_Name;  
[VB] Public Const Cube\_Name As OleDbLiteral  
[JScript] public var Cube\_Name : OleDbLiteral;

*Description*

The name of a cube in a schema (or the catalog if the provider does not support schemas).

ToString

[C#] public const OleDbLiteral Cursor\_Name;

1 [C++] public: const OleDbLiteral Cursor\_Name;  
 2 [VB] Public Const Cursor\_Name As OleDbLiteral  
 3 [JScript] public var Cursor\_Name : OleDbLiteral;

4  
 5 *Description*

6 A cursor name in a text command.

7 ToString

8  
 9 [C#] public const OleDbLiteral Dimension\_Name;  
 10 [C++] public: const OleDbLiteral Dimension\_Name;  
 11 [VB] Public Const Dimension\_Name As OleDbLiteral  
 12 [JScript] public var Dimension\_Name : OleDbLiteral;

13  
 14 *Description*

15 The name of the dimension. If a dimension is part of more than one cube,  
 16 there is one row for each cube/dimension combination.

17 ToString

18  
 19 [C#] public const OleDbLiteral Escape\_Percent\_Prefix;  
 20 [C++] public: const OleDbLiteral Escape\_Percent\_Prefix;  
 21 [VB] Public Const Escape\_Percent\_Prefix As OleDbLiteral  
 22 [JScript] public var Escape\_Percent\_Prefix : OleDbLiteral;

23  
 24 *Description*

25

The character used in a LIKE clause to escape the character returned for the DBLITERAL\_LIKE\_PERCENT literal. For example, if a percent sign (%) is used to match zero or more characters and this is a backslash (\), the characters "abc\%%" match all character values that start with "abc%". Some SQL dialects support a clause (the ESCAPE clause) that can be used to override this value.

#### ToString

```
[C#] public const OleDbLiteral Escape_Percent_Suffix;  
[C++] public: const OleDbLiteral Escape_Percent_Suffix;  
[VB] Public Const Escape_Percent_Suffix As OleDbLiteral  
[JScript] public var Escape_Percent_Suffix : OleDbLiteral;
```

#### *Description*

The escape character, if any, used to suffix the character returned for the DBLITERAL\_LIKE\_PERCENT literal. For example, if a percent sign (%) is used to match zero or more characters and percent signs are escaped by enclosing in open and close square brackets, DBLITERAL\_ESCAPE\_PERCENT\_PREFIX is "[", DBLITERAL\_ESCAPE\_PERCENT\_SUFFIX is "]", and the characters "abc[%]%" match all character values that start with "abc%". Providers that do not use a suffix character to escape the DBLITERAL\_ESCAPE\_PERCENT character do not return this literal value and can set the lt member of the DBLITERAL structure to DBLITERAL\_INVALID if requested.

#### ToString

```
[C#] public const OleDbLiteral Escape_Underscore_Prefix;
```

```

1  [C++] public: const OleDbLiteral Escape_Underscore_Prefix;
2  [VB] Public Const Escape_Underscore_Prefix As OleDbLiteral
3  [JScript] public var Escape_Underscore_Prefix : OleDbLiteral;

```

#### 5 *Description*

6       The character used in a LIKE clause to escape the character returned for the  
 7       DBLITERAL\_LIKE\_UNDERSCORE literal. For example, if an underscore ( ) is  
 8       used to match exactly one character and this is a backslash (\\), the characters  
 9       "abc\\\_" match all character values that are five characters long and start with  
 10       "abc\_". Some SQL dialects support a clause (the ESCAPE clause) that can be used  
 11       to override this value.

#### 12       ToString

```

13
14 [C#] public const OleDbLiteral Escape_Underscore_Suffix;
15 [C++] public: const OleDbLiteral Escape_Underscore_Suffix;
16 [VB] Public Const Escape_Underscore_Suffix As OleDbLiteral
17 [JScript] public var Escape_Underscore_Suffix : OleDbLiteral;

```

#### 19 *Description*

20       The character used in a LIKE clause to escape the character returned for the  
 21       DBLITERAL\_LIKE\_UNDERSCORE literal. For example, if an underscore ( ) is  
 22       used to match exactly one character and this is a backslash (\\), the characters  
 23       "abc\\\_" match all character values that are five characters long and start with  
 24       "abc\_". Some SQL dialects support a clause (the ESCAPE clause) that can be used  
 25       to override this value.

## ToString

```
[C#] public const OleDbLiteral Hierarchy_Name;
[C++] public: const OleDbLiteral Hierarchy_Name;
[VB] Public Const Hierarchy_Name As OleDbLiteral
[JScript] public var Hierarchy_Name : OleDbLiteral;
```

### *Description*

The name of the hierarchy. If the dimension does not contain a hierarchy or has only one hierarchy, the current column contains a null value.

## ToString

```
[C#] public const OleDbLiteral Index_Name;
[C++] public: const OleDbLiteral Index_Name;
[VB] Public Const Index_Name As OleDbLiteral
[JScript] public var Index_Name : OleDbLiteral;
```

### *Description*

An index name used in a text command or in a data-definition interface.

## ToString

```
[C#] public const OleDbLiteral Invalid;
[C++] public: const OleDbLiteral Invalid;
[VB] Public Const Invalid As OleDbLiteral
[JScript] public var Invalid : OleDbLiteral;
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

An invalid value.

ToString

```
[C#] public const OleDbLiteral Level_Name;
[C++] public: const OleDbLiteral Level_Name;
[VB] Public Const Level_Name As OleDbLiteral
[JScript] public var Level_Name : OleDbLiteral;
```

*Description*

Name of the cube to which the current level belongs.

ToString

```
[C#] public const OleDbLiteral Like_Percent;
[C++] public: const OleDbLiteral Like_Percent;
[VB] Public Const Like_Percent As OleDbLiteral
[JScript] public var Like_Percent : OleDbLiteral;
```

*Description*

The character used in a LIKE clause to match zero or more characters. For example, if this is a percent sign (%), the characters "abc%" match all character values that start with "abc".

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const OleDbLiteral Like_Underscore;
[C++] public: const OleDbLiteral Like_Underscore;
[VB] Public Const Like_Underscore As OleDbLiteral
[JScript] public var Like_Underscore : OleDbLiteral;
```

*Description*

The character used in a LIKE clause to match exactly one character. For example, if this is an underscore ( \_ ), the characters "abc\_" match all character values that are four characters long and start with "abc".

ToString

```
[C#] public const OleDbLiteral Member_Name;
[C++] public: const OleDbLiteral Member_Name;
[VB] Public Const Member_Name As OleDbLiteral
[JScript] public var Member_Name : OleDbLiteral;
```

*Description*

The name of the member.

ToString

```
[C#] public const OleDbLiteral Procedure_Name;
[C++] public: const OleDbLiteral Procedure_Name;
[VB] Public Const Procedure_Name As OleDbLiteral
[JScript] public var Procedure_Name : OleDbLiteral;
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

A procedure name in a text command.

ToString

```
[C#] public const OleDbLiteral Property_Name;
[C++] public: const OleDbLiteral Property_Name;
[VB] Public Const Property_Name As OleDbLiteral
[JScript] public var Property_Name : OleDbLiteral;
```

*Description*

The name of the property.

ToString

```
[C#] public const OleDbLiteral Quote_Prefix;
[C++] public: const OleDbLiteral Quote_Prefix;
[VB] Public Const Quote_Prefix As OleDbLiteral
[JScript] public var Quote_Prefix : OleDbLiteral;
```

*Description*

The character used in a text command as the opening quote for quoting  
identifiers that contain special characters.

ToString

```
[C#] public const OleDbLiteral Quote_Suffix;
```

1 [C++] public: const OleDbLiteral Quote\_Suffix;  
 2 [VB] Public Const Quote\_Suffix As OleDbLiteral  
 3 [JScript] public var Quote\_Suffix : OleDbLiteral;  
 4

5 *Description*

6 The character used in a text command as the closing quote for quoting  
 7 identifiers that contain special characters. 1.x providers that use the same character  
 8 as the prefix and suffix may not return this literal value and can set the It member  
 9 of the DBLITERAL structure to DBLITERAL\_INVALID if requested.

10 ToString

11  
 12 [C#] public const OleDbLiteral Schema\_Name;  
 13 [C++] public: const OleDbLiteral Schema\_Name;  
 14 [VB] Public Const Schema\_Name As OleDbLiteral  
 15 [JScript] public var Schema\_Name : OleDbLiteral;  
 16

17 *Description*

18 A schema name in a text command.

19 ToString

20  
 21 [C#] public const OleDbLiteral Schema\_Separator;  
 22 [C++] public: const OleDbLiteral Schema\_Separator;  
 23 [VB] Public Const Schema\_Separator As OleDbLiteral  
 24 [JScript] public var Schema\_Separator : OleDbLiteral;  
 25

1  
2 *Description*

3 The character that separates the schema name from the rest of the identifier  
4 in a text command.

5 ToString

6  
7 [C#] public const OleDbLiteral Table\_Name;

8 [C++] public: const OleDbLiteral Table\_Name;

9 [VB] Public Const Table\_Name As OleDbLiteral

10 [JScript] public var Table\_Name : OleDbLiteral;

11  
12 *Description*

13 A table name used in a text command or in a data-definition interface.

14 ToString

15  
16 [C#] public const OleDbLiteral Text\_Command;

17 [C++] public: const OleDbLiteral Text\_Command;

18 [VB] Public Const Text\_Command As OleDbLiteral

19 [JScript] public var Text\_Command : OleDbLiteral;

20  
21 *Description*

22 A text command, such as an SQL statement.

23 ToString

24  
25 [C#] public const OleDbLiteral User\_Name;

1 [C++] public: const OleDbLiteral User\_Name;  
 2 [VB] Public Const User\_Name As OleDbLiteral  
 3 [JScript] public var User\_Name : OleDbLiteral;

4  
 5 *Description*

6 A user name in a text command.

7 ToString

8  
 9 [C#] public const OleDbLiteral View\_Name;  
 10 [C++] public: const OleDbLiteral View\_Name;  
 11 [VB] Public Const View\_Name As OleDbLiteral  
 12 [JScript] public var View\_Name : OleDbLiteral;

13  
 14 *Description*

15 A view name in a text command.

16 OleDbParameter class (System.Data.OleDb)

17 ToString

18  
 19  
 20 *Description*

21 Represents a parameter to an **System.Data.OleDb.OleDbCommand** and  
 22 optionally, its mapping to a **System.Data.DataSet** column.

23 Parameter names are not case sensitive.

24 OleDbParameter

25 *Example Syntax:*

## ToString

[C#] public OleDbParameter();

[C++] public: OleDbParameter();

[VB] Public Sub New()

[JScript] public function OleDbParameter(); Initializes a new instance of the **System.Data.OleDb.OleDbParameter** class.

### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbParameter** class.

**OleDbParameter**

### *Example Syntax:*

#### ToString

[C#] public OleDbParameter(string name, object value);

[C++] public: OleDbParameter(String\* name, Object\* value);

[VB] Public Sub New(ByVal name As String, ByVal value As Object)

[JScript] public function OleDbParameter(name : String, value : Object);

### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbParameter** class with the parameter name and an **System.Data.OleDb.OleDbParameter** object. The name of the parameter to map. An **System.Data.OleDb.OleDbParameter** object.

OleDbParameter

*Example Syntax:*

ToString

[C#] public OleDbParameter(string name, DbType dataType);

[C++] public: OleDbParameter(String\* name, DbType dataType);

[VB] Public Sub New(ByVal name As String, ByVal dataType As DbType)

[JScript] public function OleDbParameter(name : String, dataType : DbType);

### Description

Initializes a new instance of the **System.Data.OleDb.OleDbParameter** class with the parameter name and data type.

The data type, and if appropriate, **System.Data.OleDb.OleDbParameter.Size** and **System.Data.OleDb.OleDbParameter.Precision** are inferred from the value of the *dataType* parameter. The name of the parameter to map. One of the **System.Data.OleDb.OleDbType** values.

OleDbParameter

*Example Syntax:*

ToString

[C#] public OleDbParameter(string name, DbType dataType, int size);

[C++] public: OleDbParameter(String\* name, DbType dataType, int size);

[VB] Public Sub New(ByVal name As String, ByVal dataType As DbType, ByVal size As Integer)

1 [JScript] public function OleDbParameter(name : String, dataType : OleDbType,  
2 size : int);

3

4 *Description*

5       Initializes a new instance of the **System.Data.OleDb.OleDbParameter**  
6 class with the parameter name, data type, and width.

7       The **System.Data.OleDb.OleDbParameter.Size** is inferred from the value  
8 of the *dataType* parameter if it is not explicitly set in the *size* parameter. The name  
9 of the parameter to map. One of the **System.Data.OleDb.OleDbType** values. The  
10 width of the parameter.

11       OleDbParameter

12       *Example Syntax:*

13       ToString

14

15 [C#] public OleDbParameter(string name, OleDbType dataType, int size, string  
16 srcColumn);

17 [C++] public: OleDbParameter(String\* name, OleDbType dataType, int size,  
18 String\* srcColumn);

19 [VB] Public Sub New(ByVal name As String, ByVal dataType As OleDbType,  
20 ByVal size As Integer, ByVal srcColumn As String)

21 [JScript] public function OleDbParameter(name : String, dataType : OleDbType,  
22 size : int, srcColumn : String);

23

24 *Description*

25

1        Initializes a new instance of the **System.Data.OleDb.OleDbParameter**  
 2        class with the parameter name, data type, width, and source column name.

3        The **System.Data.OleDb.OleDbParameter.Size** is inferred from the value  
 4        of the *dataType* parameter if it is not explicitly set in the *size* parameter. The name  
 5        of the parameter to map. One of the **System.Data.OleDb.OleDbType** values. The  
 6        width of the parameter. The name of the source column.

7        OleDbParameter

8        *Example Syntax:*

9        ToString

10  
 11        [C#] public OleDbParameter(string parameterName, OleDbType dbType, int size,  
 12        ParameterDirection direction, bool isNullable, byte precision, byte scale, string  
 13        srcColumn, DataRowVersion srcVersion, object value);

14        [C++] public: OleDbParameter(String\* parameterName, OleDbType dbType, int  
 15        size, ParameterDirection direction, bool isNullable, unsigned char precision,  
 16        unsigned char scale, String\* srcColumn, DataRowVersion srcVersion, Object\*  
 17        value);

18        [VB] Public Sub New(ByVal parameterName As String, ByVal dbType As  
 19        OleDbType, ByVal size As Integer, ByVal direction As ParameterDirection,  
 20        ByVal isNullable As Boolean, ByVal precision As Byte, ByVal scale As Byte,  
 21        ByVal srcColumn As String, ByVal srcVersion As DataRowVersion, ByVal value  
 22        As Object)

23        [JScript] public function OleDbParameter(parameterName : String, dbType :  
 24        OleDbType, size : int, direction : ParameterDirection, isNullable : Boolean,  
 25        precision : Byte, scale : Byte, srcColumn : String, srcVersion : DataRowVersion,

value : Object);

### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbParameter** class with the parameter name, data type, width, source column name, parameter direction, numeric precision, and other properties.

The **System.Data.OleDb.OleDbParameter.Size** and **System.Data.OleDb.OleDbParameter.Precision** are inferred from the value of the *dataType* parameter if they are not explicitly set in the *size* and *precision* parameters. The name of the parameter. One of the **System.Data.OleDb.OleDbType** values. The width of the parameter. One of the **System.Data.ParameterDirection** values. **true** if the value of the field can be null; otherwise, **false**. The total number of digits to the left and right of the decimal point to which **System.Data.OleDb.OleDbParameter.Value** is resolved. The total number of decimal places to which **System.Data.OleDb.OleDbParameter.Value** is resolved. The name of the source column. One of the **System.Data.DataRowVersion** values. An **System.Object** that is the value of the **System.Data.OleDb.OleDbParameter**.

DbType

ToString

[C#] public DbType DbType {get; set;}

[C++] public: \_\_property DbType get\_DbType();public: \_\_property void set\_DbType(DbType);

[VB] Public Property DbType As DbType

[JScript] public function get DbType() : DbType; public function set DbType(DbType);

*Description*

Gets or sets the **System.Data.DbType** of the parameter.

The **System.Data.OleDb.OleDbParameter.OleDbType** and **System.Data.OleDb.OleDbParameter.DbType** are linked. Therefore, setting the **System.Data.OleDb.OleDbParameter.DbType** changes the **System.Data.OleDb.OleDbParameter.OleDbType** to a supporting **System.Data.OleDb.OleDbParameter.OleDbType**.

Direction

ToString

[C#] public ParameterDirection Direction {get; set;}

[C++] public: \_\_property ParameterDirection get \_Direction(); public: \_\_property void set \_Direction(ParameterDirection);

[VB] Public Property Direction As ParameterDirection

[JScript] public function get Direction() : ParameterDirection; public function set Direction(ParameterDirection);

*Description*

Gets or sets a value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.

If the **System.Data.ParameterDirection** is output, and execution of the associated **System.Data.OleDb.OleDbCommand** does not return a value, the **System.Data.OleDb.OleDbParameter** contains a null value.

IsNullable

ToString

[C#] public bool IsNullable {get; set;}

[C++] public: \_\_property bool get\_IsNullable();public: \_\_property void set\_IsNullable(bool);

[VB] Public Property IsNullable As Boolean

[JScript] public function get IsNullable() : Boolean;public function set IsNullable(Boolean);

#### *Description*

Gets or sets a value indicating whether the parameter accepts null values.

Null values are handled using the **System.DBNull** class.

OleDbType

ToString

[C#] public OleDbType OleDbType {get; set;}

[C++] public: \_\_property OleDbType get\_OleDbType();public: \_\_property void set\_OleDbType(OleDbType);

[VB] Public Property OleDbType As OleDbType

[JScript] public function get OleDbType() : OleDbType;public function set OleDbType(OleDbType);

## Description

Gets or sets the **System.Data.OleDb.OleDbType** of the parameter.

The **System.Data.OleDb.OleDbParameter.OleDbType** and **System.Data.OleDb.OleDbParameter.DbType** are linked. Therefore, setting the **System.Data.OleDb.OleDbParameter.DbType** changes the **System.Data.OleDb.OleDbParameter.OleDbType** to a supporting **System.Data.OleDb.OleDbType**.

ParameterName

ToString

[C#] public string ParameterName {get; set;}

[C++] public: \_\_property String\* get\_ParameterName();public: \_\_property void set\_ParameterName(String\*);

[VB] Public Property ParameterName As String

[JScript] public function get ParameterName() : String;public function set ParameterName(String);

## Description

Gets or sets the name of the **System.Data.OleDb.OleDbParameter**.

The OLE DB .NET Provider uses positional parameters that are marked with a question mark (?) instead of named parameters.

Precision

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public byte Precision {get; set;}

[C++] public: __property unsigned char get_Precision();public: __property void
set_Precision(unsigned char);

[VB] Public Property Precision As Byte

[JScript] public function get Precision() : Byte;public function set Precision(Byte);
```

*Description*

Gets or sets the maximum number of digits used to represent the  
**System.Data.OleDb.OleDbParameter.Value** property.  
 The **System.Data.OleDb.OleDbParameter.Precision** property is only  
 used for decimal and numeric input parameters.

Scale  
 ToString

```
[C#] public byte Scale {get; set;}

[C++] public: __property unsigned char get_Scale();public: __property void
set_Scale(unsigned char);

[VB] Public Property Scale As Byte

[JScript] public function get Scale() : Byte;public function set Scale(Byte);
```

*Description*

Gets or sets the number of decimal places to which  
**System.Data.OleDb.OleDbParameter.Value** is resolved.

The **System.Data.OleDb.OleDbParameter.Scale** property is only used for decimal and numeric input parameters.

Size

ToString

[C#] public int Size {get; set;}

[C++] public: \_\_property int get\_Size();public: \_\_property void set\_Size(int);

[VB] Public Property Size As Integer

[JScript] public function get Size() : int;public function set Size(int);

#### *Description*

Gets or sets the maximum size, in bytes, of the data within the column.

The **System.Data.OleDb.OleDbParameter.Size** property is used for binary and string types.

SourceColumn

ToString

[C#] public string SourceColumn {get; set;}

[C++] public: \_\_property String\* get\_SourceColumn();public: \_\_property void set\_SourceColumn(String\*);

[VB] Public Property SourceColumn As String

[JScript] public function get SourceColumn() : String;public function set SourceColumn(String);

#### *Description*

1 [VB] Function Clone() As Object Implements ICloneable.Clone

2 [JScript] function ICloneable.Clone() : Object;

3 ToString

4

5 [C#] public override string ToString();

6 [C++] public: String\* ToString();

7 [VB] Overrides Public Function ToString() As String

8 [JScript] public override function ToString() : String;

9

10 *Description*

11 Gets a string containing the

12 **System.Data.OleDb.OleDbParameter.ParameterName** .

13 *Return Value:* A string containing the

14 **System.Data.OleDb.OleDbParameter.ParameterName** .

15 OleDbParameterCollection class (System.Data.OleDb)

16 ToString

17

18

19 *Description*

20 Collects all parameters relevant to an

21 **System.Data.OleDb.OleDbCommand** and their respective mappings to

22 **System.Data.DataSet** columns.

23 The number of parameters in the collection must be equal to the number of

24 parameter placeholders within the command text, or the OLE DB .NET Data

25 Provider may raise an error.

Count

ToString

[C#] public int Count {get;}

[C++] public: \_\_property int get\_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

*Description*

Gets the number of **System.Data.OleDb.OleDbParameter** objects in the collection.

Item

ToString

[C#] public OleDbParameter this[int index] {get; set;}

[C++] public: \_\_property OleDbParameter\* get\_Item(int index);public:

\_\_property void set\_Item(int index, OleDbParameter\*);

[VB] Public Default Property Item(ByVal index As Integer) As OleDbParameter

[JScript] returnValue =

OleDbParameterCollectionObject.Item(index);OleDbParameterCollectionObject.Item(index) = returnValue; Gets or sets the **System.Data.OleDb.OleDbParameter** with a specified attribute.

*Description*

Gets or sets the **System.Data.OleDb.OleDbParameter** at the specified index. The zero-based index of the parameter to retrieve.

Item

ToString

[C#] public OleDbParameter this[string parameterName] {get; set;}

[C++] public: \_\_property OleDbParameter\* get\_Item(String\* parameterName); public: \_\_property void set\_Item(String\* parameterName, OleDbParameter\*);

[VB] Public Default Property Item(ByVal parameterName As String) As OleDbParameter

[JScript] returnValue =

OleDbParameterCollectionObject.Item(parameterName); OleDbParameterCollectionObject.Item(parameterName) = returnValue;

### *Description*

Gets or sets the **System.Data.OleDb.OleDbParameter** with the specified name. The name of the parameter to retrieve.

Add

[C#] public int Add(object value);

[C++] public: \_\_sealed int Add(Object\* value);

[VB] NotOverridable Public Function Add(ByVal value As Object) As Integer

[JScript] public function Add(value : Object) : int; Adds an

**System.Data.OleDb.OleDbParameter** to the

## 1 **System.Data.OleDb.OleDbCommand .**

### 3 *Description*

4 Adds an **System.Data.OleDb.OleDbParameter** object to the  
5 **System.Data.OleDb.OleDbCommand** .

6 *Return Value:* A reference to the new **System.Data.OleDb.OleDbParameter**  
7 object. The **System.Data.OleDb.OleDbParameter** object to add to the collection.

#### 8 **Add**

10 [C#] public OleDbParameter Add(OleDbParameter value);

11 [C++] public: OleDbParameter\* Add(OleDbParameter\* value);

12 [VB] Public Function Add(ByVal value As OleDbParameter) As OleDbParameter

13 [JScript] public function Add(value : OleDbParameter) : OleDbParameter;

### 15 *Description*

16 Adds the specified **System.Data.OleDb.OleDbParameter** to the  
17 **System.Data.OleDb.OleDbCommand** .

18 *Return Value:* A reference to the new **System.Data.OleDb.OleDbParameter**  
19 object. The **System.Data.OleDb.OleDbParameter** to add to the collection.

#### 20 **Add**

22 [C#] public OleDbParameter Add(string parameterName, object value);

23 [C++] public: OleDbParameter\* Add(String\* parameterName, Object\* value);

24 [VB] Public Function Add(ByVal parameterName As String, ByVal value As  
25 Object) As OleDbParameter

```

1 [JScript] public function Add(parameterName : String, value : Object) :
2 OleDbParameter;
    
```

#### *Description*

Adds an **System.Data.OleDb.OleDbParameter** to the **System.Data.OleDb.OleDbCommand** given the parameter name and value.

*Return Value:* A reference to the new **System.Data.OleDb.OleDbParameter** object.

When you specify **System.DBNull.Value** in the *value* parameter, you should also explicitly set the **System.Data.OleDb.OleDbType** as demonstrated in this C# example: `OleDbCommand rComm = new OleDbCommand(null, rConn); rComm.CommandText = "insert into mytable values (?)"; rComm.Parameters.Add("@p1", DBNull.Value); rComm.Parameters["@p1"].OleDbType = OleDbType.Integer;` The **System.Data.OleDb.OleDbParameter.Value** of the **System.Data.OleDb.OleDbParameter** to add to the collection.

#### Add

```

19 [C#] public OleDbParameter Add(string parameterName, OleDbType
20 OleDbType);
    
```

```

21 [C++] public: OleDbParameter* Add(String* parameterName, OleDbType
22 OleDbType);
    
```

```

23 [VB] Public Function Add(ByVal parameterName As String, ByVal OleDbType
24 As OleDbType) As OleDbParameter
    
```

```

25 [JScript] public function Add(parameterName : String, OleDbType : OleDbType) :
    
```

OleDbParameter;

*Description*

Adds an **System.Data.OleDb.OleDbParameter** to the **System.Data.OleDb.OleDbCommand** given the parameter name and data type.

*Return Value:* A reference to the new **System.Data.OleDb.OleDbParameter** object.

**Add**

[C#] public OleDbParameter Add(string parameterName, OleDbType oleDbType, int size);

[C++] public: OleDbParameter\* Add(String\* parameterName, OleDbType oleDbType, int size);

[VB] Public Function Add(ByVal parameterName As String, ByVal oleDbType As OleDbType, ByVal size As Integer) As OleDbParameter

[JScript] public function Add(parameterName : String, oleDbType : OleDbType, size : int) : OleDbParameter;

*Description*

Adds an **System.Data.OleDb.OleDbParameter** to the **System.Data.OleDb.OleDbCommand** given the the parameter name, data type, and column width.

*Return Value:* A reference to the new **System.Data.OleDb.OleDbParameter** object. The width of the column.

**Add**

```

1
2 [C#] public OleDbParameter Add(string parameterName, OleDbType oleDbType,
3 int size, string sourceColumn);
4 [C++] public: OleDbParameter* Add(String* parameterName, OleDbType
5 oleDbType, int size, String* sourceColumn);
6 [VB] Public Function Add(ByVal parameterName As String, ByVal oleDbType
7 As OleDbType, ByVal size As Integer, ByVal sourceColumn As String) As
8 OleDbParameter
9 [JScript] public function Add(parameterName : String, oleDbType : OleDbType,
10 size : int, sourceColumn : String) : OleDbParameter;
11

```

## 12 *Description*

13 Adds an **System.Data.OleDb.OleDbParameter** to the  
14 **System.Data.OleDb.OleDbCommand** given the parameter name, data type,  
15 column width, and source column name.

16 *Return Value:* A reference to the new **System.Data.OleDb.OleDbParameter**  
17 object. The width of the column. The name of the source column.

## 18 *Clear*

```

19
20 [C#] public void Clear();
21 [C++] public: __sealed void Clear();
22 [VB] NotOverridable Public Sub Clear()
23 [JScript] public function Clear();
24

```

## 25 *Description*

1 Removes all items from the collection.

2 Contains

3

4 [C#] public bool Contains(object value);

5 [C++] public: \_\_sealed bool Contains(Object\* value);

6 [VB] NotOverridable Public Function Contains(ByVal value As Object) As

7 Boolean

8 [JScript] public function Contains(value : Object) : Boolean;

9

10 *Description*

11 Gets a value indicating whether an **System.Data.OleDb.OleDbParameter**  
12 object exists in the collection.

13 *Return Value:* **true** if the collection contains the

14 **System.Data.OleDb.OleDbParameter** ; otherwise, **false** . The value of the

15 **System.Data.OleDb.OleDbParameter** object to find.

16 Contains

17

18 [C#] public bool Contains(string value);

19 [C++] public: \_\_sealed bool Contains(String\* value);

20 [VB] NotOverridable Public Function Contains(ByVal value As String) As

21 Boolean

22 [JScript] public function Contains(value : String) : Boolean; Indicates whether an

23 **System.Data.OleDb.OleDbParameter** exists in the collection.

24

25 *Description*

Gets a value indicating whether an **System.Data.OleDb.OleDbParameter** with the specified parameter name exists in the collection.

*Return Value:* **true** if the collection contains the parameter; otherwise, **false**. The name of the parameter.

#### CopyTo

[C#] public void CopyTo(Array array, int index);

[C++] public: \_\_sealed void CopyTo(Array\* array, int index);

[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)

[JScript] public function CopyTo(array : Array, index : int);

#### Description

Copies **System.Data.OleDb.OleDbParameter** objects from the **System.Data.OleDb.OleDbParameterCollection** to the specified array. The **System.Array** into which to copy the **System.Data.OleDb.OleDbParameter** objects. The starting index of the array.

#### GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: \_\_sealed IEnumerator\* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

#### Description

## IndexOf

[C#] public int IndexOf(object value);

[C++] public: \_\_sealed int IndexOf(Object\* value);

[VB] NotOverridable Public Function IndexOf(ByVal value As Object) As Integer

[JScript] public function IndexOf(value : Object) : int;

### *Description*

Gets the location of the **System.Data.OleDb.OleDbParameter** object in the collection.

**Return Value:** The location of the **System.Data.OleDb.OleDbParameter** in the collection. The **System.Data.OleDb.OleDbParameter** object to locate.

## IndexOf

[C#] public int IndexOf(string parameterName);

[C++] public: \_\_sealed int IndexOf(String\* parameterName);

[VB] NotOverridable Public Function IndexOf(ByVal parameterName As String)

As Integer

[JScript] public function IndexOf(parameterName : String) : int; Gets the location of the **System.Data.OleDb.OleDbParameter** in the collection.

### *Description*

Gets the location of the **System.Data.OleDb.OleDbParameter** in the collection with the specified parameter name.

*Return Value:* The location of the **System.Data.OleDb.OleDbParameter** in the collection.

### Insert

[C#] public void Insert(int index, object value);

[C++] public: \_\_sealed void Insert(int index, Object\* value);

[VB] NotOverridable Public Sub Insert(ByVal index As Integer, ByVal value As Object)

[JScript] public function Insert(index : int, value : Object);

### Description

Inserts an **System.Data.OleDb.OleDbParameter** in the collection at the specified index. The zero-based index where the parameter is to be inserted within the collection. The **System.Data.OleDb.OleDbParameter** to add to the collection.

### Remove

[C#] public void Remove(object value);

[C++] public: \_\_sealed void Remove(Object\* value);

[VB] NotOverridable Public Sub Remove(ByVal value As Object)

[JScript] public function Remove(value : Object);

### Description

Removes the specified **System.Data.OleDb.OleDbParameter** from the collection. The **System.Data.OleDb.OleDbParameter** object to remove from the collection.

**RemoveAt**

[C#] public void RemoveAt(int index);

[C++] public: \_\_sealed void RemoveAt(int index);

[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)

[JScript] public function RemoveAt(index : int); Removes the specified **System.Data.OleDb.OleDbParameter** from the collection.

#### *Description*

Removes the **System.Data.OleDb.OleDbParameter** at the specified index from the collection. The zero-based index of the parameter to remove.

**RemoveAt**

[C#] public void RemoveAt(string parameterName);

[C++] public: \_\_sealed void RemoveAt(String\* parameterName);

[VB] NotOverridable Public Sub RemoveAt(ByVal parameterName As String)

[JScript] public function RemoveAt(parameterName : String);

#### *Description*

Removes the **System.Data.OleDb.OleDbParameter** with the specified name from the collection.

**OleDbPermission** class (System.Data.OleDb)

ToString

*Description*

Provides the capability for the OLE DB .NET Data Provider to ensure that a user has a security level adequate to access an OLE DB data source.

OleDbPermission

*Example Syntax:*

ToString

[C#] public OleDbPermission();

[C++] public: OleDbPermission();

[VB] Public Sub New()

[JScript] public function OleDbPermission(); Initializes a new instance of the **System.Data.OleDb.OleDbPermission** class.

*Description*

Initializes a new instance of the **System.Data.OleDb.OleDbPermission** class.

OleDbPermission

*Example Syntax:*

ToString

[C#] public OleDbPermission(PermissionState state);

[C++] public: OleDbPermission(PermissionState state);

1 [VB] Public Sub New(ByVal state As PermissionState)  
 2 [JScript] public function OleDbPermission(state : PermissionState);

3  
 4 *Description*

5 One of the **System.Security.Permissions.PermissionState** values.  
 6 OleDbPermission

7 *Example Syntax:*  
 8 ToString

9  
 10 [C#] public OleDbPermission(PermissionState state, bool allowBlankPassword);

11 [C++] public: OleDbPermission(PermissionState state, bool  
 12 allowBlankPassword);

13 [VB] Public Sub New(ByVal state As PermissionState, ByVal  
 14 allowBlankPassword As Boolean)

15 [JScript] public function OleDbPermission(state : PermissionState,  
 16 allowBlankPassword : Boolean);

17  
 18 *Description*

19 One of the **System.Security.Permissions.PermissionState** values.  
 20 Indicates whether a blank password is allowed.

21 AllowBlankPassword  
 22 Provider  
 23 ToString

24  
 25

### *Description*

Gets or sets a comma-delimited list of providers allowed by the security policy.

#### Copy

[C#] public override IPPermission Copy();

[C++] public: IPPermission\* Copy();

[VB] Overrides Public Function Copy() As IPPermission

[JScript] public override function Copy() : IPPermission;

### *Description*

#### FromXml

[C#] public override void FromXml(SecurityElement securityElement);

[C++] public: void FromXml(SecurityElement\* securityElement);

[VB] Overrides Public Sub FromXml(ByVal securityElement As SecurityElement)

[JScript] public override function FromXml(securityElement : SecurityElement);

### *Description*

#### Intersect

[C#] public override IPPermission Intersect(IPPermission target);

[C++] public: IPPermission\* Intersect(IPPermission\* target);

[VB] Overrides Public Function Intersect(ByVal target As IPPermission) As

IPPermission

[JScript] public override function Intersect(target : IPPermission) : IPPermission;

### *Description*

IsSubsetOf

[C#] public override bool IsSubsetOf(IPPermission target);

[C++] public: bool IsSubsetOf(IPPermission\* target);

[VB] Overrides Public Function IsSubsetOf(ByVal target As IPPermission) As

Boolean

[JScript] public override function IsSubsetOf(target : IPPermission) : Boolean;

### *Description*

ToXml

[C#] public override SecurityElement ToXml();

[C++] public: SecurityElement\* ToXml();

[VB] Overrides Public Function ToXml() As SecurityElement

[JScript] public override function ToXml() : SecurityElement;

### *Description*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

Union

[C#] public override IPPermission Union(IPPermission target);

[C++] public: IPPermission\* Union(IPPermission\* target);

[VB] Overrides Public Function Union(ByVal target As IPPermission) As IPPermission

[JScript] public override function Union(target : IPPermission) : IPPermission;

*Description*

OleDbPermissionAttribute class (System.Data.OleDb)

Union

*Description*

Associates a security action with a custom security attribute.

OleDbPermissionAttribute

*Example Syntax:*

Union

[C#] public OleDbPermissionAttribute(SecurityAction action);

[C++] public: OleDbPermissionAttribute(SecurityAction action);

[VB] Public Sub New(ByVal action As SecurityAction)

[JScript] public function OleDbPermissionAttribute(action : SecurityAction);

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Initializes a new instance of the **System.Data.OleDb.OleDbPermissionAttribute** class.

*Return Value:* An **System.Data.OleDb.OleDbPermissionAttribute** object. One of the the **System.Security.Permissions.SecurityAction** values representing an action that can be performed using declarative security.

- Action
- AllowBlankPassword
- Provider
- Union

*Description*

Gets or sets a comma-delimited string containing a list of supported providers.

- TypeId
- Unrestricted
- CreatePermission

```
[C#] public override IPPermission CreatePermission();  
[C++] public: IPPermission* CreatePermission();  
[VB] Overrides Public Function CreatePermission() As IPPermission  
[JScript] public override function CreatePermission() : IPPermission;
```

1  
2 *Description*

3 Returns an **System.Data.OleDb.OleDbPermission** object that is  
4 configured according to the attribute properties.

5 *Return Value:* An **System.Data.OleDb.OleDbPermission** object.

6 OleDbRowUpdatedEventArgs class (System.Data.OleDb)

7 ToString

8  
9  
10 *Description*

11 Provides data for the  
12 **System.Data.OleDb.OleDbDataAdapter.RowUpdated** event.

13 The **System.Data.OleDb.OleDbDataAdapter.RowUpdated** event is  
14 raised when an  
15 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to a  
16 row is completed.

17 OleDbRowUpdatedEventArgs

18 *Example Syntax:*

19 ToString

20  
21 [C#] public OleDbRowUpdatedEventArgs(DataRow dataRow, IDbCommand  
22 command, StatementType statementType, DataTableMapping tableMapping);  
23 [C++] public: OleDbRowUpdatedEventArgs(DataRow\* dataRow, IDbCommand\*  
24 command, StatementType statementType, DataTableMapping\* tableMapping);  
25 [VB] Public Sub New(ByVal dataRow As DataRow, ByVal command As

```

1 IDbCommand, ByVal statementType As StatementType, ByVal tableMapping As
2 DataTableMapping)
3 [JScript] public function OleDbRowUpdatedEventArgs(dataRow : DataRow,
4 command : IDbCommand, statementType : StatementType, tableMapping :
5 DataTableMapping);

```

### *Description*

Initializes a new instance of the **System.Data.OleDb.OleDbRowUpdatedEventArgs** class. The **System.Data.DataRow** sent through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**. The **System.Data.IDbCommand** executed when **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** is called. One of the **System.Data.StatementType** values that specifies the type of query executed. The **System.Data.Common.DataTableMapping** sent through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**.

Command

ToString

```

20 [C#] public new OleDbCommand Command {get;}
21 [C++] public: __property OleDbCommand* get_Command();
22 [VB] Public ReadOnly Property Command As OleDbCommand
23 [JScript] public function get Command() : OleDbCommand;

```

### *Description*

1 Gets the **System.Data.OleDb.OleDbCommand** executed when  
2 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** is  
3 called.

4 Errors

5 RecordsAffected

6 Row

7 StatementType

8 Status

9 TableMapping

10 OleDbRowUpdatedEventHandler delegate (System.Data.OleDb)

11 ToString

12  
13  
14 *Description*

15 Represents the method that will handle the  
16 **System.Data.OleDb.OleDbDataAdapter.RowUpdated** event of an  
17 **System.Data.OleDb.OleDbDataAdapter** . The source of the event. The  
18 **System.Data.OleDb.OleDbRowUpdatedEventArgs** that contains the event data.

19 The handler is not required perform any action, and your code should avoid  
20 generating exceptions or allowing exceptions to propagate to the calling method.  
21 Any exceptions that do reach the caller are ignored.

22 OleDbRowUpdatingEventArgs class (System.Data.OleDb)

23 ToString

1  
2  
3 *Description*

4 Provides data for the  
5 **System.Data.OleDb.OleDbDataAdapter.RowUpdating** event.

6 The **System.Data.OleDb.OleDbDataAdapter.RowUpdating** event is  
7 raised before an  
8 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to a  
9 row.

10 OleDbRowUpdatingEventArgs

11 *Example Syntax:*

12 ToString

13  
14 [C#] public OleDbRowUpdatingEventArgs(DataRow dataRow, IDbCommand  
15 command, StatementType statementType, DataTableMapping tableMapping);

16 [C++] public: OleDbRowUpdatingEventArgs(DataRow\* dataRow,  
17 IDbCommand\* command, StatementType statementType, DataTableMapping\*  
18 tableMapping);

19 [VB] Public Sub New(ByVal dataRow As DataRow, ByVal command As  
20 IDbCommand, ByVal statementType As StatementType, ByVal tableMapping As  
21 DataTableMapping)

22 [JScript] public function OleDbRowUpdatingEventArgs(dataRow : DataRow,  
23 command : IDbCommand, statementType : StatementType, tableMapping :  
24 DataTableMapping);

## Description

Initializes a new instance of the **System.Data.OleDb.OleDbRowUpdatingEventArgs** class. The **System.Data.DataRow** to **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** . The **System.Data.IDbCommand** to execute during **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** . One of the **System.Data.StatementType** values that specifies the type of query executed. The **System.Data.Common.DataTableMapping** sent through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**.

Command

ToString

[C#] public new OleDbCommand Command {get; set;}

[C++] public: \_\_property OleDbCommand\* get\_Command();public: \_\_property void set\_Command(OleDbCommand\*);

[VB] Public Property Command As OleDbCommand

[JScript] public function get Command() : OleDbCommand;public function set Command(OleDbCommand);

## Description

Gets or sets the **System.Data.OleDb.OleDbCommand** to execute when performing the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** .

1

2

3

4

5

6

7

8

9

## 10

11

12

13

14

16

18

19

20

21

22

## 23

24

25

1 Returns the type of schema table specified by the  
2 **System.Data.OleDb.OleDbConnection.GetOleDbSchemaTable(System.Guid,**  
3 **System.Object[])** method.

4 Each field in the **System.Data.OleDb.OleDbSchemaGuid** class maps to  
5 an OLE DB schema rowset. For more information, see Appendix B: Schema  
6 rowsets in the OLE DB Programmer's Reference.

7 ToString

8  
9 [C#] public static readonly Guid Assertions;  
10 [C++] public: static Guid Assertions;  
11 [VB] Public Shared ReadOnly Assertions As Guid  
12 [JScript] public static var Assertions : Guid;

13  
14 *Description*

15 Returns the assertions defined in the catalog that are owned by a given user.  
16 **System.Data.OleDb.OleDbSchemaGuid.Assertions** maps to the OLE DB  
17 ASSERTIONS rowset. Unless otherwise specified, restriction columns are  
18 returned in the following order.

19 ToString

20  
21 [C#] public static readonly Guid Catalogs;  
22 [C++] public: static Guid Catalogs;  
23 [VB] Public Shared ReadOnly Catalogs As Guid  
24 [JScript] public static var Catalogs : Guid;

## Description

Returns the physical attributes associated with catalogs accessible from the data source. Returns the assertions defined in the catalog that are owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Catalogs** maps to the OLE DB CATALOGS rowset. Unless otherwise specified, restriction columns are returned in the following order.

### ToString

[C#] public static readonly Guid Character\_Sets;

[C++] public: static Guid Character\_Sets;

[VB] Public Shared ReadOnly Character\_Sets As Guid

[JScript] public static var Character\_Sets : Guid;

## Description

Returns the character sets defined in the catalog that are accessible to a given user.

**System.Data.OleDb.OleDbSchemaGuid.Character\_Sets** maps to the OLE DB CHARACTER\_SETS rowset. Unless otherwise specified, restriction columns are returned in the following order.

### ToString

[C#] public static readonly Guid Check\_Constraints;

[C++] public: static Guid Check\_Constraints;

1 [VB] Public Shared ReadOnly Check\_Constraints As Guid

2 [JScript] public static var Check\_Constraints : Guid;

3

4 *Description*

5 Returns the check constraints defined in the catalog that are owned by a  
6 given user.

7 **System.Data.OleDb.OleDbSchemaGuid.Check\_Constraints** maps to the  
8 OLE DB CHECK\_CONSTRAINTS rowset. Unless otherwise specified,  
9 restriction columns are returned in the following order.

10 ToString

11

12 [C#] public static readonly Guid Check\_Constraints\_By\_Table;

13 [C++] public: static Guid Check\_Constraints\_By\_Table;

14 [VB] Public Shared ReadOnly Check\_Constraints\_By\_Table As Guid

15 [JScript] public static var Check\_Constraints\_By\_Table : Guid;

16

17 *Description*

18 Returns the check constraints defined in the catalog that are owned by a  
19 given user.

20 **System.Data.OleDb.OleDbSchemaGuid.Check\_Constraints** maps to the  
21 OLE DB CHECK\_CONSTRAINTS rowset. Unless otherwise specified,  
22 restriction columns are returned in the following order.

23 ToString

24

25 [C#] public static readonly Guid Collations;

[C++] public: static Guid Collations;

[VB] Public Shared ReadOnly Collations As Guid

[JScript] public static var Collations : Guid;

### *Description*

Returns the character collations defined in the catalog that are accessible to a given user.

**System.Data.OleDb.OleDbSchemaGuid.Collations** maps to the OLE DB COLLATIONS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Column\_Domain\_Usage;

[C++] public: static Guid Column\_Domain\_Usage;

[VB] Public Shared ReadOnly Column\_Domain\_Usage As Guid

[JScript] public static var Column\_Domain\_Usage : Guid;

### *Description*

Returns the columns defined in the catalog that are dependent on a domain defined in the catalog and owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Column\_Domain\_Usage** maps to the OLE DB COLUMN\_DOMAIN\_USAGE rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Column\_Privileges;

[C++] public: static Guid Column\_Privileges;

[VB] Public Shared ReadOnly Column\_Privileges As Guid

[JScript] public static var Column\_Privileges : Guid;

### *Description*

Returns the privileges on columns of tables defined in the catalog that are available to, or granted by, a given user.

**System.Data.OleDb.OleDbSchemaGuid.Column\_Privileges** maps to the OLE DB COLUMN\_PRIVILEGES rowset. Unless otherwise specified, restriction columns are returned in the following order.

### *ToString*

[C#] public static readonly Guid Columns;

[C++] public: static Guid Columns;

[VB] Public Shared ReadOnly Columns As Guid

[JScript] public static var Columns : Guid;

### *Description*

Returns the columns of tables (including views) defined in the catalog that are accessible to a given user.

**System.Data.OleDb.OleDbSchemaGuid.Columns** maps to the OLE DB COLUMNS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

```
[C#] public static readonly Guid Constraint_Column_Usage;
[C++] public: static Guid Constraint_Column_Usage;
[VB] Public Shared ReadOnly Constraint_Column_Usage As Guid
[JScript] public static var Constraint_Column_Usage : Guid;
```

*Description*

Returns the columns used by referential constraints, unique constraints, check constraints, and assertions, defined in the catalog and owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Constraint\_Column\_Usage** maps to the OLE DB CONSTRAINT\_COLUMN\_USAGE rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

```
[C#] public static readonly Guid Constraint_Table_Usage;
[C++] public: static Guid Constraint_Table_Usage;
[VB] Public Shared ReadOnly Constraint_Table_Usage As Guid
[JScript] public static var Constraint_Table_Usage : Guid;
```

*Description*

Returns the tables that are used by referential constraints, unique constraints, check constraints, and assertions defined in the catalog and owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Constraint\_Table\_Usage** maps to the OLE DB CONSTRAINT\_TABLE\_USAGE rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

```
[C#] public static readonly Guid DbInfoLiterals;
[C++] public: static Guid DbInfoLiterals;
[VB] Public Shared ReadOnly DbInfoLiterals As Guid
[JScript] public static var DbInfoLiterals : Guid;
```

#### *Description*

Returns a list of provider-specific literals used in text commands.

Using **System.Data.OleDb.OleDbSchemaGuid.DbInfoLiterals** is equivalent to calling the OLE DB IDBInfo::GetLiteralInfo interface, or the ADO **Connection.OpenSchema** method with the **adSchemaDBInfoLiterals** constant.

ToString

```
[C#] public static readonly Guid Foreign_Keys;
[C++] public: static Guid Foreign_Keys;
[VB] Public Shared ReadOnly Foreign_Keys As Guid
[JScript] public static var Foreign_Keys : Guid;
```

#### *Description*

Returns the foreign key columns defined in the catalog by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Foreign\_Keys** maps to the OLE DB FOREIGN\_KEYS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Indexes;  
 [C++] public: static Guid Indexes;  
 [VB] Public Shared ReadOnly Indexes As Guid  
 [JScript] public static var Indexes : Guid;

#### *Description*

Returns the indexes defined in the catalog that are owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Indexes** maps to the OLE DB INDEXES rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Key\_Column\_Usage;  
 [C++] public: static Guid Key\_Column\_Usage;  
 [VB] Public Shared ReadOnly Key\_Column\_Usage As Guid  
 [JScript] public static var Key\_Column\_Usage : Guid;

#### *Description*

Returns the columns defined in the catalog that are constrained as keys by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Key\_Column\_Usage** maps to the OLE DB KEY\_COLUMN rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Primary\_Keys;

[C++] public: static Guid Primary\_Keys;

[VB] Public Shared ReadOnly Primary\_Keys As Guid

[JScript] public static var Primary\_Keys : Guid;

#### *Description*

Returns the primary key columns defined in the catalog by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Primary\_Keys** maps to the OLE DB PRIMARY\_KEYS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Procedure\_Columns;

[C++] public: static Guid Procedure\_Columns;

[VB] Public Shared ReadOnly Procedure\_Columns As Guid

[JScript] public static var Procedure\_Columns : Guid;

#### *Description*

Returns information about the columns of rowsets returned by procedures.

**System.Data.OleDb.OleDbSchemaGuid.Procedure\_Columns** maps to the OLE DB PROCEDURE\_COLUMNS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Procedure\_Parameters;

[C++] public: static Guid Procedure\_Parameters;

[VB] Public Shared ReadOnly Procedure\_Parameters As Guid

[JScript] public static var Procedure\_Parameters : Guid;

#### *Description*

Returns information about the parameters and return codes of procedures.

**System.Data.OleDb.OleDbSchemaGuid.Procedure\_Parameters** maps to the OLE DB PROCEDURE\_PARAMETERS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Procedures;

[C++] public: static Guid Procedures;

[VB] Public Shared ReadOnly Procedures As Guid

[JScript] public static var Procedures : Guid;

#### *Description*

Returns the procedures defined in the catalog that are owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Procedures** maps to the OLE DB PROCEDURES rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Provider\_Types;

[C++] public: static Guid Provider\_Types;

[VB] Public Shared ReadOnly Provider\_Types As Guid

[JScript] public static var Provider\_Types : Guid;

*Description*

Returns the base data types supported by the .NET data provider.

**System.Data.OleDb.OleDbSchemaGuid.Provider\_Types** maps to the OLE DB PROVIDER\_TYPES rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Referential\_Constraints;

[C++] public: static Guid Referential\_Constraints;

[VB] Public Shared ReadOnly Referential\_Constraints As Guid

[JScript] public static var Referential\_Constraints : Guid;

*Description*

Returns the referential constraints defined in the catalog that are owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Referential\_Constraints** maps to the OLE DB REFERENTIAL\_CONSTRAINTS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Schemata;

[C++] public: static Guid Schemata;

[VB] Public Shared ReadOnly Schemata As Guid

[JScript] public static var Schemata : Guid;

#### *Description*

Returns the schema objects that are owned by a given user.

**System.Data.OleDb.OleDbSchemaGuid.Schemata** maps to the OLE DB SCHEMATAS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Sql\_Languages;

[C++] public: static Guid Sql\_Languages;

[VB] Public Shared ReadOnly Sql\_Languages As Guid

[JScript] public static var Sql\_Languages : Guid;

#### *Description*

Returns the conformance levels, options, and dialects supported by the SQL-implementation processing data defined in the catalog.

1       **System.Data.OleDb.OleDbSchemaGuid.Sql\_Languages** maps to the  
2 OLE DB SQL\_LANGUAGES rowset. Unless otherwise specified, restriction  
3 columns are returned in the following order.

4       ToString

5  
6       [C#] public static readonly Guid Statistics;

7       [C++] public: static Guid Statistics;

8       [VB] Public Shared ReadOnly Statistics As Guid

9       [JScript] public static var Statistics : Guid;

10  
11       *Description*

12       Returns the statistics defined in the catalog that are owned by a given user.

13       **System.Data.OleDb.OleDbSchemaGuid.Statistics** maps to the OLE DB  
14 STATISTICS rowset. Unless otherwise specified, restriction columns are returned  
15 in the following order.

16       ToString

17  
18       [C#] public static readonly Guid Table\_Constraints;

19       [C++] public: static Guid Table\_Constraints;

20       [VB] Public Shared ReadOnly Table\_Constraints As Guid

21       [JScript] public static var Table\_Constraints : Guid;

22  
23       *Description*

24       Returns the table constraints defined in the catalog that are owned by a  
25 given user.

**System.Data.OleDb.OleDbSchemaGuid.Table\_Constraints** maps to the OLE DB TABLE\_CONSTRAINTS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Table\_Privileges;

[C++] public: static Guid Table\_Privileges;

[VB] Public Shared ReadOnly Table\_Privileges As Guid

[JScript] public static var Table\_Privileges : Guid;

#### *Description*

Returns the privileges on tables defined in the catalog that are available to, or granted by, a given user.

**System.Data.OleDb.OleDbSchemaGuid.Table\_Privileges** maps to the OLE DB TABLE\_PRIVILEGES rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Table\_Statistics;

[C++] public: static Guid Table\_Statistics;

[VB] Public Shared ReadOnly Table\_Statistics As Guid

[JScript] public static var Table\_Statistics : Guid;

#### *Description*

Describes the available set of statistics on tables in the provider.

**System.Data.OleDb.OleDbSchemaGuid.Table\_Statistics** maps to the OLE DB TABLE\_STATISTICS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Tables;  
[C++] public: static Guid Tables;  
[VB] Public Shared ReadOnly Tables As Guid  
[JScript] public static var Tables : Guid;

*Description*

Returns the tables (including views) defined in the catalog that are accessible to a given user.

**System.Data.OleDb.OleDbSchemaGuid.Tables** maps to the OLE DB TABLES rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Tables\_Info;  
[C++] public: static Guid Tables\_Info;  
[VB] Public Shared ReadOnly Tables\_Info As Guid  
[JScript] public static var Tables\_Info : Guid;

*Description*

Returns the tables (including views) that are accessible to a given user.

**System.Data.OleDb.OleDbSchemaGuid.Tables\_Info** maps to the OLE DB TABLES\_INFO rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Translations;

[C++] public: static Guid Translations;

[VB] Public Shared ReadOnly Translations As Guid

[JScript] public static var Translations : Guid;

### *Description*

Returns the character translations defined in the catalog that are accessible to a given user.

**System.Data.OleDb.OleDbSchemaGuid.Translations** maps to the OLE DB TRANSLATIONS rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Trustee;

[C++] public: static Guid Trustee;

[VB] Public Shared ReadOnly Trustee As Guid

[JScript] public static var Trustee : Guid;

### *Description*

Identifies the trustees defined in the data source.

**System.Data.OleDb.OleDbSchemaGuid.Trustee** maps to the OLE DB TRUSTEES schema. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid Usage\_Privileges;  
 [C++] public: static Guid Usage\_Privileges;  
 [VB] Public Shared ReadOnly Usage\_Privileges As Guid  
 [JScript] public static var Usage\_Privileges : Guid;

#### *Description*

Returns the USAGE privileges on objects defined in the catalog that are available to, or granted by, a given user.

**System.Data.OleDb.OleDbSchemaGuid.Usage\_Privileges** maps to the OLE DB USAGE\_PRIVILEGES rowset. Unless otherwise specified, restriction columns are returned in the following order.

ToString

[C#] public static readonly Guid View\_Column\_Usage;  
 [C++] public: static Guid View\_Column\_Usage;  
 [VB] Public Shared ReadOnly View\_Column\_Usage As Guid  
 [JScript] public static var View\_Column\_Usage : Guid;

#### *Description*

Returns the columns on which viewed tables, defined in the catalog and owned by a given user, are dependent.

**System.Data.OleDb.OleDbSchemaGuid.View\_Column\_Usage** maps to the OLE DB VIEW\_COLUMN\_USAGE rowset. Unless otherwise specified, restriction columns are returned in the following order.

**ToString**

[C#] public static readonly Guid View\_Table\_Usage;

[C++] public: static Guid View\_Table\_Usage;

[VB] Public Shared ReadOnly View\_Table\_Usage As Guid

[JScript] public static var View\_Table\_Usage : Guid;

### *Description*

Returns the tables on which viewed tables, defined in the catalog and owned by a given user, are dependent.

**System.Data.OleDb.OleDbSchemaGuid.View\_Table\_Usage** maps to the OLE DB VIEW\_TABLE\_USAGE rowset. Unless otherwise specified, restriction columns are returned in the following order.

**ToString**

[C#] public static readonly Guid Views;

[C++] public: static Guid Views;

[VB] Public Shared ReadOnly Views As Guid

[JScript] public static var Views : Guid;

1  
2 *Description*

3 Returns the views defined in the catalog that are accessible to a given user.

4 **System.Data.OleDb.OleDbSchemaGuid.Views** maps to the OLE DB  
5 VIEWS rowset. Unless otherwise specified, restriction columns are returned in the  
6 following order.

7 OleDbSchemaGuid

8 *Example Syntax:*

9 ToString

10  
11 [C#] public OleDbSchemaGuid();

12 [C++] public: OleDbSchemaGuid();

13 [VB] Public Sub New()

14 [JScript] public function OleDbSchemaGuid();

15 OleDbTransaction class (System.Data.OleDb)

16 ToString

17  
18  
19 *Description*

20 Represents an SQL transaction to be made at a data source.

21 The application creates an **System.Data.OleDb.OleDbTransaction** object  
22 by calling **System.Data.SqlClient.SqlConnection.BeginTransaction** on the  
23 **System.Data.OleDb.OleDbConnection** object. All subsequent operations  
24 associated with the transaction (for example, committing or aborting the  
25

1 transaction), are performed on the **System.Data.OleDb.OleDbTransaction**  
2 object.

3       **Connection**

4       **ToString**

6 [C#] public OleDbConnection Connection {get;}

7 [C++] public: \_\_property OleDbConnection\* get\_Connection();

8 [VB] Public ReadOnly Property Connection As OleDbConnection

9 [JScript] public function get Connection() : OleDbConnection;

10       **IsolationLevel**

11       **ToString**

13 [C#] public IsolationLevel IsolationLevel {get;}

14 [C++] public: \_\_property IsolationLevel get\_IsolationLevel();

15 [VB] Public ReadOnly Property IsolationLevel As IsolationLevel

16 [JScript] public function get IsolationLevel() : IsolationLevel;

18 *Description*

19       Specifies the **System.Data.IsolationLevel** for this transaction.

20       Parallel transactions are not supported. Therefore, the  
21 **System.Data.IsolationLevel** applies to the entire transaction.

22       **Begin**

24 [C#] public OleDbTransaction Begin();

25 [C++] public: OleDbTransaction\* Begin();

1 [VB] Public Function Begin() As OleDbTransaction

2 [JScript] public function Begin() : OleDbTransaction;

4 *Description*

5 Initiates a nested database transaction.

6 The new transaction is nested within the current transaction.

7 Begin

9 [C#] public OleDbTransaction Begin(IsolationLevel isolevel);

10 [C++] public: OleDbTransaction\* Begin(IsolationLevel isolevel);

11 [VB] Public Function Begin(ByVal isolevel As IsolationLevel) As

12 OleDbTransaction

13 [JScript] public function Begin(isolevel : IsolationLevel) : OleDbTransaction;

14 Initiates a nested database transaction.

16 *Description*

17 Initiates a nested database transaction and specifies the isolation level to  
18 use for the new transaction.

19 The new transaction is nested within the current transaction. The isolation  
20 level to use for the transaction.

21 Commit

23 [C#] public void Commit();

24 [C++] public: \_\_sealed void Commit();

25 [VB] NotOverridable Public Sub Commit()

1 [JScript] public function Commit();

2

3 *Description*

4 Commits the database transaction.

5 Finalize

6

7 [C#] ~OleDbTransaction();

8 [C++] ~OleDbTransaction();

9 [VB] Overrides Protected Sub Finalize()

10 [JScript] protected override function Finalize();

11

12 *Description*

13 Frees resources before the **System.Data.OleDb.OleDbTransaction** is  
14 reclaimed by the Garbage Collector.

15 Rollback

16

17 [C#] public void Rollback();

18 [C++] public: \_\_sealed void Rollback();

19 [VB] NotOverridable Public Sub Rollback()

20 [JScript] public function Rollback();

21

22 *Description*

23 Rolls back a transaction from a pending state.

24 The transaction can only be rolled back from a pending state (after

25 **System.Data.OleDb.OleDbConnection.BeginTransaction(System.Data.Isolati**

1 **onLevel)** has been called, but before  
 2 **System.Data.OleDb.OleDbTransaction.Commit** is called).

3 **IDisposable.Dispose**

4  
 5 [C#] void IDisposable.Dispose();

6 [C++] void IDisposable::Dispose();

7 [VB] Sub Dispose() Implements IDisposable.Dispose

8 [JScript] function IDisposable.Dispose();

9 **OleDbType** enumeration (System.Data.OleDb)

10 **ToString**

11  
 12  
 13 *Description*

14 Gets the data type of a field, a property, or an  
 15 **System.Data.OleDb.OleDbParameter** . The corresponding OLE DB data type is  
 16 shown in parentheses in the description of each member.

17 **ToString**

18  
 19 [C#] public const OleDbType BigInt;

20 [C++] public: const OleDbType BigInt;

21 [VB] Public Const BigInt As OleDbType

22 [JScript] public var BigInt : OleDbType;

23  
 24 *Description*

25 A 64-bit signed integer (DBTYPE\_I8). This maps to **System.Int64** .

ToString

[C#] public const OleDbType Binary;  
[C++] public: const OleDbType Binary;  
[VB] Public Const Binary As OleDbType  
[JScript] public var Binary : OleDbType;

*Description*

A stream of binary data (DBTYPE\_BYTES). This maps to an **System.Array** of type **System.Byte**.

ToString

[C#] public const OleDbType Boolean;  
[C++] public: const OleDbType Boolean;  
[VB] Public Const Boolean As OleDbType  
[JScript] public var Boolean : OleDbType;

*Description*

A boolean value (DBTYPE\_BOOL). This maps to **System.Boolean**.

ToString

[C#] public const OleDbType BSTR;  
[C++] public: const OleDbType BSTR;  
[VB] Public Const BSTR As OleDbType  
[JScript] public var BSTR : OleDbType;

1  
2 *Description*

3 A null-terminated character string of Unicode characters  
4 (DBTYPE\_BSTR). This maps to **System.String** .

5 ToString

6  
7 [C#] public const OleDbType Char;

8 [C++] public: const OleDbType Char;

9 [VB] Public Const Char As OleDbType

10 [JScript] public var Char : OleDbType;

11  
12 *Description*

13 A character string (DBTYPE\_STR). This maps to **System.String** .

14 ToString

15  
16 [C#] public const OleDbType Currency;

17 [C++] public: const OleDbType Currency;

18 [VB] Public Const Currency As OleDbType

19 [JScript] public var Currency : OleDbType;

20  
21 *Description*

22 A currency value ranging from -2 (or -922,337,203,685,477.5808) to 2 -1  
23 (or +922,337,203,685,477.5807) with an accuracy to a ten-thousandth of a  
24 currency unit (DBTYPE\_CY). This maps to **System.Decimal** .

25 ToString

```

1
2 [C#] public const OleDbType Date;
3 [C++] public: const OleDbType Date;
4 [VB] Public Const Date As OleDbType
5 [JScript] public var Date : OleDbType;
6

```

#### *Description*

Date data, stored as a double (DBTYPE\_DATE). The whole portion is the number of days since December 30, 1899, while the fractional portion is a fraction of a day. This maps to **System.DateTime**.

#### *ToString*

```

12
13 [C#] public const OleDbType DBDate;
14 [C++] public: const OleDbType DBDate;
15 [VB] Public Const DBDate As OleDbType
16 [JScript] public var DBDate : OleDbType;
17

```

#### *Description*

Date data in the format yyyyymmdd (DBTYPE\_DBDATE). This maps to **System.DateTime**.

#### *ToString*

```

22
23 [C#] public const OleDbType DBTime;
24 [C++] public: const OleDbType DBTime;
25 [VB] Public Const DBTime As OleDbType

```

1 [JScript] public var DBTime : OleDbType;

2

3 *Description*

4 Time data in the format hhmmss (DBTYPE\_DBTIME). This maps to

5 **System.TimeSpan** .

6 ToString

7

8 [C#] public const OleDbType DBTimeStamp;

9 [C++] public: const OleDbType DBTimeStamp;

10 [VB] Public Const DBTimeStamp As OleDbType

11 [JScript] public var DBTimeStamp : OleDbType;

12

13 *Description*

14 Data and time data in the format yyyyymmddhhmmss

15 (DBTYPE\_DBTIMESTAMP). This maps to **System.DateTime** .

16 ToString

17

18 [C#] public const OleDbType Decimal;

19 [C++] public: const OleDbType Decimal;

20 [VB] Public Const Decimal As OleDbType

21 [JScript] public var Decimal : OleDbType;

22

23 *Description*

24 A fixed precision and scale numeric value between -10 -1 and 10 -1

25 (DBTYPE\_DECIMAL). This maps to **System.Decimal** .



*Description*

A 32-bit error code (DBTYPE\_ERROR). This maps to **System.Exception** .

ToString

[C#] public const OleDbType Filetime;

[C++] public: const OleDbType Filetime;

[VB] Public Const Filetime As OleDbType

[JScript] public var Filetime : OleDbType;

*Description*

A 64-bit unsigned integer representing the number of 100-nanosecond intervals since January 1, 1601 (DBTYPE\_FILETIME). This maps to **System.DateTime** .

ToString

[C#] public const OleDbType Guid;

[C++] public: const OleDbType Guid;

[VB] Public Const Guid As OleDbType

[JScript] public var Guid : OleDbType;

*Description*

A globally unique identifier (or GUID) (DBTYPE\_GUID). This maps to **System.Guid** .

ToString

```

1
2 [C#] public const OleDbType IDispatch;
3 [C++] public: const OleDbType IDispatch;
4 [VB] Public Const IDispatch As OleDbType
5 [JScript] public var IDispatch : OleDbType;
6

```

*Description*

A pointer to an IDispatch interface (DBTYPE\_IDISPATCH). This maps to

**System.Object .**

ToString

```

11
12 [C#] public const OleDbType Integer;
13 [C++] public: const OleDbType Integer;
14 [VB] Public Const Integer As OleDbType
15 [JScript] public var Integer : OleDbType;
16

```

*Description*

A 32-bit signed integer (DBTYPE\_I4). This maps to **System.Int32 .**

ToString

```

17
18
19
20
21 [C#] public const OleDbType IUnknown;
22 [C++] public: const OleDbType IUnknown;
23 [VB] Public Const IUnknown As OleDbType
24 [JScript] public var IUnknown : OleDbType;
25

```



```

1
2 [C#] public const OleDbType LongVarChar;
3 [C++] public: const OleDbType LongVarChar;
4 [VB] Public Const LongVarChar As OleDbType
5 [JScript] public var LongVarChar : OleDbType;

```

#### *Description*

A long null-terminated Unicode string value (**System.Data.OleDb.OleDbParameter** only). This maps to **System.String** .

**ToString**

```

12 [C#] public const OleDbType Numeric;
13 [C++] public: const OleDbType Numeric;
14 [VB] Public Const Numeric As OleDbType
15 [JScript] public var Numeric : OleDbType;

```

#### *Description*

An exact numeric value with a fixed precision and scale (**DBTYPE\_NUMERIC**). This maps to **System.Decimal** .

**ToString**

```

22 [C#] public const OleDbType PropVariant;
23 [C++] public: const OleDbType PropVariant;
24 [VB] Public Const PropVariant As OleDbType
25 [JScript] public var PropVariant : OleDbType;

```

*Description*

An automation PROPVARIANT (DBTYPE\_PROP\_VARIANT). This maps to **System.Object** .

ToString

[C#] public const OleDbType Single;

[C++] public: const OleDbType Single;

[VB] Public Const Single As OleDbType

[JScript] public var Single : OleDbType;

*Description*

A floating point number within the range of -3.40E +38 through 3.40E +38 (DBTYPE\_R4). This maps to **System.Single** .

ToString

[C#] public const OleDbType SmallInt;

[C++] public: const OleDbType SmallInt;

[VB] Public Const SmallInt As OleDbType

[JScript] public var SmallInt : OleDbType;

*Description*

A 16-bit signed integer (DBTYPE\_I2). This maps to **System.Int16** .

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const OleDbType TinyInt;
[C++] public: const OleDbType TinyInt;
[VB] Public Const TinyInt As OleDbType
[JScript] public var TinyInt : OleDbType;
```

*Description*

A 8-bit signed integer (DBTYPE\_I1). This maps to **System.SByte** .  
ToString

```
[C#] public const OleDbType UnsignedBigInt;
[C++] public: const OleDbType UnsignedBigInt;
[VB] Public Const UnsignedBigInt As OleDbType
[JScript] public var UnsignedBigInt : OleDbType;
```

*Description*

A 64-bit unsigned integer (DBTYPE\_UI8). This maps to **System.UInt64** .  
ToString

```
[C#] public const OleDbType UnsignedInt;
[C++] public: const OleDbType UnsignedInt;
[VB] Public Const UnsignedInt As OleDbType
[JScript] public var UnsignedInt : OleDbType;
```

*Description*

A 32-bit unsigned integer (DBTYPE\_UI4). This maps to **System.UInt32** .

ToString

[C#] public const OleDbType UnsignedSmallInt;

[C++] public: const OleDbType UnsignedSmallInt;

[VB] Public Const UnsignedSmallInt As OleDbType

[JScript] public var UnsignedSmallInt : OleDbType;

*Description*

A 16-bit unsigned integer (DBTYPE\_UI2). This maps to **System.UInt16** .

ToString

[C#] public const OleDbType UnsignedTinyInt;

[C++] public: const OleDbType UnsignedTinyInt;

[VB] Public Const UnsignedTinyInt As OleDbType

[JScript] public var UnsignedTinyInt : OleDbType;

*Description*

A 8-bit unsigned integer (DBTYPE\_UI1). This maps to **System.Byte** .

ToString

[C#] public const OleDbType VarBinary;

[C++] public: const OleDbType VarBinary;

[VB] Public Const VarBinary As OleDbType

[JScript] public var VarBinary : OleDbType;

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

### Description

A variable-length stream of binary data ( **System.Data.OleDb.OleDbParameter** only). This maps to an **System.Array** of type **System.Byte** .

## ToString

[C#] public const OleDbType VarChar;  
[C++] public: const OleDbType VarChar;  
[VB] Public Const VarChar As OleDbType  
[JScript] public var VarChar : OleDbType;

### Description

A variable-length stream of non-Unicode characters (**System.Data.OleDb.OleDbParameter** only). This maps to **System.String**.

## ToString

[C#] public const OleDbType Variant;  
[C++] public: const OleDbType Variant;  
[VB] Public Const Variant As OleDbType  
[JScript] public var Variant : OleDbType;

### Description

1 A special data type that can contain numeric, string, binary, or date data, as  
2 well as the special values Empty and Null (DBTYPE\_VARIANT). This type is  
3 assumed if no other is specified. This maps to **System.Object** .

4 ToString

5  
6 [C#] public const OleDbType VarNumeric;  
7 [C++] public: const OleDbType VarNumeric;  
8 [VB] Public Const VarNumeric As OleDbType  
9 [JScript] public var VarNumeric : OleDbType;

10  
11 *Description*

12 A variable-length numeric value ( **System.Data.OleDb.OleDbParameter**  
13 only). This maps to **System.Decimal** .

14 ToString

15  
16 [C#] public const OleDbType VarWChar;  
17 [C++] public: const OleDbType VarWChar;  
18 [VB] Public Const VarWChar As Ol

19  
20  
21  
22 **System.Data.SqlClient**

23 *Description*

24 The **System.Data.SqlClient** namespace is the SQL Server .NET Data  
25 Provider.

## SqlClientPermission class (System.Data.SqlClient)

### *Description*

Provides the capability for the SQL Server .NET Data Provider to ensure that a user has a security level adequate to access a data source.

#### Constructors:

SqlClientPermission

#### *Example Syntax:*

```
[C#]                public                SqlClientPermission();
[C++]                public:                SqlClientPermission();
[VB]                Public                Sub                New()
[JScript] public function SqlClientPermission(); Initializes a new instance of the
System.Data.SqlClient.SqlClientPermission class.
```

### *Description*

Initializes a new instance of the System.Data.SqlClient.SqlClientPermission class.

SqlClientPermission

#### *Example Syntax:*

```
[C#]                public                SqlClientPermission(PermissionState                state);
[C++]                public:                SqlClientPermission(PermissionState                state);
[VB]                Public                Sub                New(ByVal                state                As                PermissionState)
```

1 [JScript] public function SqlClientPermission(state : PermissionState);

2

3 *Description*

4 One of the **System.Security.Permissions.PermissionState** values.

5 **SqlClientPermission**

6 *Example Syntax:*

7

8 [C#] public SqlClientPermission(PermissionState state, bool  
9 allowBlankPassword);

10 [C++] public: SqlClientPermission(PermissionState state, bool  
11 allowBlankPassword);

12 [VB] Public Sub New(ByVal state As PermissionState, ByVal  
13 allowBlankPassword As Boolean)

14 [JScript] public function SqlClientPermission(state : PermissionState,  
15 allowBlankPassword : Boolean);

16

17 *Description*

18 One of the **System.Security.Permissions.PermissionState** values.

19 Indicates whether a blank password is allowed.

20 Properties:

21 AllowBlankPassword

22 Methods:

23 **SqlClientPermissionAttribute** class (System.Data.SqlClient)

24 Union

25

1  
2  
3 *Description*

4 Associates a security action with a custom security attribute.

5 `SqlClientPermissionAttribute`

6 *Example Syntax:*

7 Union

9 [C#] public SqlClientPermissionAttribute(SecurityAction action);

10 [C++] public: SqlClientPermissionAttribute(SecurityAction action);

11 [VB] Public Sub New(ByVal action As SecurityAction)

12 [JScript] public function SqlClientPermissionAttribute(action : SecurityAction);

13  
14 *Description*

15 Initializes a new instance of the  
16 **System.Data.SqlClient.SqlClientPermissionAttribute** class.

17 *Return Value:* A **System.Data.SqlClient.SqlClientPermissionAttribute** object.

18 One of the the **System.Security.Permissions.SecurityAction** values representing  
19 an action that can be performed using declarative security.

20 Action

21 AllowBlankPassword

22 TypeId

23 Unrestricted

24 CreatePermission

```

1
2 [C#]      public      override      IPermission      CreatePermission();
3 [C++]      public:      IPermission*      CreatePermission();
4 [VB]  Overrides  Public  Function  CreatePermission()  As  IPermission
5 [JScript]  public  override  function  CreatePermission()  :  IPermission;
6

```

### 7 *Description*

8 Returns a **System.Data.SqlClient.SqlClientPermission** object that is  
9 configured according to the attribute properties.

10 *Return Value:* A **System.Data.SqlClient.SqlClientPermission** object.

11 SqlCommand class (System.Data.SqlClient)

12 ToString

### 15 *Description*

16 Represents a Transact-SQL statement or stored procedure to execute at a  
17 SQL Server database. This class cannot be inherited.

18 When an instance of **System.Data.SqlClient.SqlCommand** is created, the  
19 read/write properties are set to their initial values. For a list of these values, see the  
20 **System.Data.SqlClient.SqlCommand** constructor.

21 SqlCommand

22 *Example Syntax:*

23 ToString

```

24
25 [C#]      public      SqlCommand();

```

```

1  [C++]                public:                SqlCommand();
2  [VB]                  Public                  Sub                  New()
3  [JScript] public function SqlCommand(); Initializes a new instance of the
4  System.Data.SqlClient.SqlCommand                class.

```

#### 6 *Description*

7        Initializes a new instance of the **System.Data.SqlClient.SqlCommand**  
8 class.

9        The following table shows initial property values for an instance of  
10 **System.Data.SqlClient.SqlCommand** .

11        SqlCommand

#### 12 *Example Syntax:*

13        ToString

```

15 [C#]                public                SqlCommand(string                cmdText);
16 [C++]                public:                SqlCommand(String*                cmdText);
17 [VB]      Public      Sub      New(ByVal      cmdText      As      String)
18 [JScript] public      function      SqlCommand(cmdText      :      String);

```

#### 20 *Description*

21        Initializes a new instance of the **System.Data.SqlClient.SqlCommand**  
22 class with the text of the query.

23        When an instance of **System.Data.SqlClient.SqlCommand** is created, the  
24 following read/write properties are set to initial values. The text of the query.

25        SqlCommand

*Example Syntax:*

ToString

```
[C#] public SqlCommand(string cmdText, SqlConnection connection);
[C++] public: SqlCommand(String* cmdText, SqlConnection* connection);
[VB] Public Sub New(ByVal cmdText As String, ByVal connection As
SqlConnection)
[JScript] public function SqlCommand(cmdText : String, connection :
```

*Description*

Initializes a new instance of the **System.Data.SqlClient.SqlCommand** class with the text of the query and a **System.Data.SqlClient.SqlConnection** .

The following table shows initial property values for an instance of **System.Data.SqlClient.SqlCommand** . The text of the query. A **System.Data.SqlClient.SqlConnection** that represents the connection to an instance of SQL Server.

SqlCommand

*Example Syntax:*

ToString

```
[C#] public SqlCommand(string cmdText, SqlConnection connection,
SqlTransaction transaction);
[C++] public: SqlCommand(String* cmdText, SqlConnection* connection,
SqlTransaction* transaction);
```

```
1  [VB] Public Sub New(ByVal cmdText As String, ByVal connection As
2  SqlConnection, ByVal transaction As SqlTransaction)
3  [JScript] public function SqlCommand(cmdText : String, connection :
4  SqlConnection, transaction : SqlTransaction);
```

### Description

Initializes a new instance of the **System.Data.SqlClient.SqlCommand** class with the text of the query, a **System.Data.SqlClient.SqlConnection** , and the **System.Data.SqlClient.Transaction** .

The following table shows initial property values for an instance of **System.Data.SqlClient.SqlCommand** . The text of the query. A **System.Data.SqlClient.SqlConnection** that represents the connection to an instance of SQL Server. The **System.Data.SqlClient.SqlTransaction** in which the **System.Data.SqlClient.SqlCommand** executes.

## CommandText

ToString

18	[C#]	public	string	CommandText	{get;	set;}
19	[C++]	public:	__property	String*	get_CommandText();	public: __property void
20					set_CommandText(String*);	
21	[VB]	Public	Property	CommandText	As	String
22	[JScript]	public	function	get	CommandText() :	String;public
23					function set	CommandText(String);

### Description

1 Gets or sets the Transact-SQL statement or stored procedure to execute at  
2 the data source.

3 When the **System.Data.SqlClient.SqlCommand.CommandType** property  
4 is set to **StoredProcedure**, the  
5 **System.Data.SqlClient.SqlCommand.CommandText** property should be set to  
6 the name of the stored procedure. The command executes this stored procedure  
7 when you call one of the Execute methods.

8 **CommandTimeout**

9 **ToString**

10  
11 [C#] public int CommandTimeout {get; set;}

12 [C++] public: \_\_property int get\_CommandTimeout();public: \_\_property void  
13 set\_CommandTimeout(int);

14 [VB] Public Property CommandTimeout As Integer

15 [JScript] public function get CommandTimeout() : int;public function set  
16 CommandTimeout(int);

17  
18 *Description*

19 Gets or sets the wait time before terminating the attempt to execute a  
20 command and generating an error.

21 A value of 0 indicates no limit, and should be avoided in a  
22 **System.Data.OleDb.OleDbCommand.CommandTimeout** because an attempt to  
23 execute a command will wait indefinitely.

24 **CommandType**

25 **ToString**

```

1
2 [C#]      public      CommandType      CommandType      {get;      set;}
3 [C++] public: __property CommandType get_CommandType();public: __property
4 void                                             set_CommandType(CommandType);
5 [VB]      Public      Property      CommandType      As      CommandType
6 [JScript] public function get CommandType() : CommandType;public function set
7 CommandType(CommandType);

```

### *Description*

Gets or sets a value indicating how the **System.Data.SqlClient.SqlCommand.CommandText** property is to be interpreted.

When you set the **System.Data.SqlClient.SqlCommand.CommandType** property to **StoredProcedure** , you should set the **System.Data.SqlClient.SqlCommand.CommandText** property to the name of the stored procedure. The command executes this stored procedure when you call one of the Execute methods.

Connection

ToString

```

21 [C#]      public      SqlConnection      Connection      {get;      set;}
22 [C++] public: __property SqlConnection* get_Connection();public: __property
23 void                                             set_Connection(SqlConnection*);
24 [VB]      Public      Property      Connection      As      SqlConnection
25 [JScript] public function get Connection() : SqlConnection;public function set

```

1 Connection(SqlConnection);

2  
3 *Description*

4 Gets or sets the **System.Data.SqlClient.SqlConnection** used by this  
5 instance of the **System.Data.SqlClient.SqlCommand** .

6 If you set **System.Data.SqlClient.SqlCommand.Connection** while a  
7 transaction is in progress and the  
8 **System.Data.SqlClient.SqlCommand.Transaction** property is not null, an  
9 **System.InvalidOperationException** is generated. If the  
10 **System.Data.SqlClient.SqlCommand.Transaction** property is not null and the  
11 transaction has already been committed or rolled back,  
12 **System.Data.SqlClient.SqlCommand.Transaction** is set to null.

13 Container

14 DesignMode

15 DesignTimeVisible

16 ToString

17  
18  
19 *Description*

20 Gets or sets a value indicating whether the command object should be  
21 visible in a Windows Forms Designer control.

22 Events

23 Parameters

24 ToString

1  
2  
3 *Description*

4 Gets the **System.Data.SqlClient.SqlParameterCollection** .

5 The SQL Server .NET Data Provider does not support the question mark  
6 (?) placeholder for passing parameters to a SQL Statement or a stored procedure  
7 called by a Command of CommandType.Text. In this case, named parameters  
8 must be used. For example: SELECT \* FROM Customers WHERE CustomerID =  
9 @CustomerID For more information see .

10 Site

11 Transaction

12 ToString  
13  
14

15 *Description*

16 Gets or sets the transaction in which the  
17 **System.Data.SqlClient.SqlCommand** executes.

18 UpdatedRowSource

19 ToString  
20

21 [C#] public UpdateRowSource UpdatedRowSource {get; set;}

22 [C++] public: \_\_property UpdateRowSource get\_UpdatedRowSource();public:

23 \_\_property void set\_UpdatedRowSource(UpdateRowSource);

24 [VB] Public Property UpdatedRowSource As UpdateRowSource

25 [JScript] public function get UpdatedRowSource() : UpdateRowSource;public

1 function set UpdatedRowSource(UpdateRowSource);

2

3 *Description*

4 Gets or sets how command results are applied to the  
5 **System.Data.DataRow** when used by the  
6 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** method  
7 of the **System.Data.Common.DbDataAdapter** .

8 Cancel

9

10 [C#] public void Cancel();

11 [C++] public: \_\_sealed void Cancel();

12 [VB] NotOverridable Public Sub Cancel()

13 [JScript] public function Cancel();

14

15 *Description*

16 Cancels the execution of a **System.Data.SqlClient.SqlCommand** .

17 CreateParameter

18

19 [C#] public SqlParameter CreateParameter();

20 [C++] public: SqlParameter\* CreateParameter();

21 [VB] Public Function CreateParameter() As SqlParameter

22 [JScript] public function CreateParameter() : SqlParameter;

23

24 *Description*

25

Creates a new instance of a **System.Data.SqlClient.SqlParameter** object.

*Return Value:* A **System.Data.SqlClient.SqlParameter** object.

The **System.Data.SqlClient.SqlCommand.CreateCommand.CreateParameter** method is a strongly-typed version of **System.Data.IDbCommand.CreateCommand.CreateParameter**.

#### ExecuteNonQuery

```
[C#]          public          int          ExecuteNonQuery();
[C++]          public:          __sealed          int          ExecuteNonQuery();
[VB]  NotOverridable  Public  Function  ExecuteNonQuery()  As  Integer
[JScript]       public       function       ExecuteNonQuery()       :       int;
```

#### Description

Executes a Transact-SQL statement against the **System.Data.SqlClient.SqlCommand.Connection** and returns the number of rows affected.

*Return Value:* The number of rows affected.

You can use the **System.Data.SqlClient.SqlCommand.ExecuteNonQuery** to perform catalog operations (for example, querying the structure of a database or creating database objects such as tables), or to change the data in a database without using a **System.Data.DataSet** by executing UPDATE, INSERT, or DELETE statements.

#### ExecuteReader

```
[C#]          public          SqlDataReader          ExecuteReader();
[C++]          public:          SqlDataReader*          ExecuteReader();
```

1 [VB] Public Function ExecuteReader() As SqlDataReader  
 2 [JScript] public function ExecuteReader() : SqlDataReader; Sends the  
 3 **System.Data.SqlClient.SqlCommand.CommandText** to the  
 4 **System.Data.SqlClient.SqlCommand.Connection** and builds a  
 5 **System.Data.SqlClient.SqlDataReader** .

### 6 *Description*

8 Sends the **System.Data.SqlClient.SqlCommand.CommandText** to the  
 9 **System.Data.SqlClient.SqlCommand.Connection** and builds a  
 10 **System.Data.SqlClient.SqlDataReader** .

11 *Return Value:* A **System.Data.SqlClient.SqlDataReader** object.

12 When the **System.Data.SqlClient.SqlCommand.CommandType** property  
 13 is set to **StoredProcedure** , the  
 14 **System.Data.SqlClient.SqlCommand.CommandText** property should be set to  
 15 the name of the stored procedure. The command executes this stored procedure  
 16 when you call **System.Data.SqlClient.SqlCommand.ExecuteReader** .

17 ExecuteReader

18  
 19 [C#] public SqlDataReader ExecuteReader(CommandBehavior behavior);  
 20 [C++] public: SqlDataReader\* ExecuteReader(CommandBehavior behavior);  
 21 [VB] Public Function ExecuteReader(ByVal behavior As CommandBehavior) As  
 22 SqlDataReader  
 23 [JScript] public function ExecuteReader(behavior : CommandBehavior) :  
 24 SqlDataReader;

25

## Description

Sends the **System.Data.SqlClient.SqlCommand.CommandText** to the **System.Data.SqlClient.SqlCommand.Connection** , and builds a **System.Data.SqlClient.SqlDataReader** using one of the **System.Data.CommandBehavior** values.

**Return Value:** A **System.Data.SqlClient.SqlDataReader** object.

When the **System.Data.SqlClient.SqlCommand.CommandType** property is set to **StoredProcedure** , the **System.Data.SqlClient.SqlCommand.CommandText** property should be set to the name of the stored procedure. The command executes this stored procedure when you call **System.Data.SqlClient.SqlCommand.ExecuteReader** . One of the **System.Data.CommandBehavior** values.

## ExecuteScalar

```
[C#]          public          object          ExecuteScalar();
[C++]          public:          __sealed          Object*          ExecuteScalar();
[VB]  NotOverridable  Public  Function  ExecuteScalar()  As  Object
[JScript]      public          function          ExecuteScalar()          :          Object;
```

## Description

Executes the query, and returns the first column of the first row in the resultset returned by the query. Extra columns or rows are ignored.

**Return Value:** The first column of the first row in the resultset.

Use the **System.Data.SqlClient.SqlCommand.ExecuteScalar** method to retrieve a single value (for example, an aggregate value) from a database. This requires less code than using the **System.Data.SqlClient.SqlCommand.ExecuteReader** method, and then performing the operations necessary to generate the single value using the data returned by a **System.Data.SqlClient.SqlDataReader**.

#### ExecuteXmlReader

```
[C#]          public          XmlReader          ExecuteXmlReader();
[C++]          public:          XmlReader*          ExecuteXmlReader();
[VB]    Public    Function    ExecuteXmlReader()    As    XmlReader
[JScript]    public    function    ExecuteXmlReader()    :    XmlReader;
```

#### Description

Sends the **System.Data.SqlClient.SqlCommand.CommandText** to the **System.Data.SqlClient.SqlCommand.Connection** and builds an **System.Xml.XmlReader** object.

*Return Value:* An **System.Xml.XmlReader** object.

The **System.Data.SqlClient.SqlCommand.CommandText** property usually specifies a Transact-SQL statement with a valid FOR XML clause. However, **System.Data.SqlClient.SqlCommand.CommandText** can also specify a statement that returns **ntext** data containing valid XML.

#### Prepare

```
[C#]          public          void          Prepare();
```

1	[C++]	public:	__sealed	void	Prepare();
2	[VB]	NotOverridable	Public	Sub	Prepare()
3	[JScript]	public	function		Prepare();

4

5 *Description*

6       Creates a prepared version of the command on an instance of SQL Server.

7       If the **System.Data.SqlClient.SqlCommand.CommandType** property is

8 set to **TableDirect** , **System.Data.SqlClient.SqlCommand.Prepare** does

9 nothing. If **System.Data.SqlClient.SqlCommand.CommandType** is set to

10 **StoredProcedure** , the call to **System.Data.SqlClient.SqlCommand.Prepare**

11 should succeed, although it may result in a no-op.

12       ResetCommandTimeout

14	[C#]	public	void	ResetCommandTimeout();
15	[C++]	public:	void	ResetCommandTimeout();
16	[VB]	Public	Sub	ResetCommandTimeout()
17	[JScript]	public	function	ResetCommandTimeout();

18

19 *Description*

20       Resets the **System.Data.SqlClient.SqlCommand.CommandTimeout**

21 property to its default value.

22       The default value of the

23 **System.Data.SqlClient.SqlCommand.CommandTimeout** is 30 seconds.

24       IDbCommand.CreateParameter

25

1			
2	[C#]	IDbDataParameter	IDbCommand.CreateParameter();
3	[C++]	IDbDataParameter*	IDbCommand::CreateParameter();
4	[VB]	Function CreateParameter() As IDbDataParameter Implements	
5		IDbCommand.CreateParameter	
6	[JScript]	function IDbCommand.CreateParameter() : IDbDataParameter;	
7		IDbCommand.ExecuteReader	
8			
9	[C#]	IDataReader	IDbCommand.ExecuteReader();
10	[C++]	IDataReader*	IDbCommand::ExecuteReader();
11	[VB]	Function ExecuteReader() As IDataReader Implements	
12		IDbCommand.ExecuteReader	
13	[JScript]	function IDbCommand.ExecuteReader() : IDataReader;	
14		IDbCommand.ExecuteReader	
15			
16	[C#]	IDataReader IDbCommand.ExecuteReader(CommandBehavior behavior);	
17	[C++]	IDataReader* IDbCommand::ExecuteReader(CommandBehavior	
18		behavior);	
19	[VB]	Function ExecuteReader(ByVal behavior As CommandBehavior) As	
20		IDataReader Implements IDbCommand.ExecuteReader	
21	[JScript]	function IDbCommand.ExecuteReader(behavior : CommandBehavior) :	
22		IDataReader;	
23		ICloneable.Clone	
24			
25	[C#]	object	ICloneable.Clone();

```

1  [C++]          Object*          ICloneable::Clone();
2  [VB]   Function Clone() As Object Implements ICloneable.Clone
3  [JScript] function ICloneable.Clone() : Object;
4          SqlCommandBuilder class (System.Data.SqlClient)
5          ToString
6
7
8  Description
9
10     Provides a means of automatically generating single-table commands used
11     to reconcile changes made to a System.Data.DataSet with the associated SQL
12     Server database. This class cannot be inherited.
13
14     The System.Data.SqlClient.SqlDataAdapter does not automatically
15     generate the Transact-SQL statements required to reconcile changes made to a
16     System.Data.DataSet with the associated instance of SQL Server. However, you
17     can create a System.Data.SqlClient.SqlCommandBuilder object to
18     automatically generate Transact-SQL statements for single-table updates if you set
19     the System.Data.SqlClient.SqlDataAdapter.SelectCommand property of the
20     System.Data.SqlClient.SqlDataAdapter . Then, any additional Transact-SQL
21     statements that you do not set are generated by the
22     System.Data.SqlClient.SqlCommandBuilder .
23
24     Example Syntax:
25
26     ToString
27
28
29  [C#]          public          SqlCommandBuilder();

```

```

1  [C++]          public:          SqlCommandBuilder();
2  [VB]           Public           Sub           New()
3  [JScript] public function SqlCommandBuilder(); Initializes a new instance of the
4  System.Data.SqlClient.SqlCommandBuilder class.

```

5  
6 *Description*

7        Initializes        a        new        instance        of        the  
8 **System.Data.SqlClient.SqlCommandBuilder** class.

9        SqlCommandBuilder

10        *Example Syntax:*

11        ToString

```

12
13 [C#]      public      SqlCommandBuilder(SqlDataAdapter      adapter);
14 [C++]     public:     SqlCommandBuilder(SqlDataAdapter*     adapter);
15 [VB]     Public     Sub     New(ByVal     adapter     As     SqlDataAdapter)
16 [JScript] public function SqlCommandBuilder(adapter : SqlDataAdapter);

```

17  
18 *Description*

19        Initializes        a        new        instance        of        the  
20 **System.Data.SqlClient.SqlCommandBuilder** class with the associated  
21 **System.Data.SqlClient.SqlDataAdapter** object. The name of the  
22 **System.Data.SqlClient.SqlDataAdapter**.

23        Container

24        DataAdapter

25        ToString

### Description

Gets or sets a **System.Data.SqlClient.SqlDataAdapter** object for which Transact-SQL statements are automatically generated.

The **System.Data.SqlClient.SqlCommandBuilder** registers itself as a listener for **System.Data.SqlClient.SqlDataAdapter.RowUpdating** events generated by the **System.Data.SqlClient.SqlDataAdapter**.

DesignMode

Events

QuotePrefix

ToString

### Description

Gets or sets the beginning character or characters to use when specifying SQL Server object names, (for example, tables or columns), that contain characters such as spaces.

Database objects in instances of SQL Server 2000 and SQL Server version 7.0 can contain any valid Microsoft Windows NT® or Microsoft Windows® 2000 characters, including spaces, commas, and semicolons. To accommodate this capability, use the **System.Data.SqlClient.SqlCommandBuilder.QuotePrefix** and **System.Data.SqlClient.SqlCommandBuilder.QuoteSuffix** properties to specify delimiters such as a left bracket and a right bracket to encapsulate the object name.

```

1      QuoteSuffix
2      ToString
3
4      [C#]      public      string      QuoteSuffix      {get;      set;}
5      [C++] public: __property String* get_QuoteSuffix();public: __property void
6      set_QuoteSuffix(String*);
7      [VB]      Public      Property      QuoteSuffix      As      String
8      [JScript] public function get QuoteSuffix() : String;public function set
9      QuoteSuffix(String);
10
11     Description
12
13         Gets or sets the ending character or characters to use when specifying SQL
14         Server object names, (for example, tables or columns), that contain characters such
15         as spaces.
16
17         Database objects in instances of SQL Server 2000 and SQL Server version
18         7.0 can contain any valid Microsoft Windows NT® or Microsoft Windows®
19         2000 characters, including spaces, commas, and semicolons. To accommodate this
20         capability, use the System.Data.SqlClient.SqlCommandBuilder.QuotePrefix
21         and System.Data.SqlClient.SqlCommandBuilder.QuoteSuffix properties to
22         specify delimiters such as a left bracket and a right bracket to encapsulate the
23         object name.
24
25         Site
26
27         Dispose
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```

103 [C#]      protected      override      void      Dispose(bool      disposing);

```

1 [C++] protected: void Dispose(bool disposing);  
 2 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)  
 3 [JScript] protected override function Dispose(disposing : Boolean); Releases the  
 4 resources used by the **System.Data.SqlClient.SqlCommandBuilder** .

#### 6 *Description*

7 Releases the unmanaged resources used by the  
 8 **System.Data.SqlClient.SqlCommandBuilder** and optionally releases the  
 9 managed resources.

10 This method is called by the public method and the  
 11 **System.Object.Finalize** method. **true** to release both managed and unmanaged  
 12 resources; **false** to release only unmanaged resources.

#### 13 **GetDeleteCommand**

15 [C#] public SqlCommand GetDeleteCommand();  
 16 [C++] public: SqlCommand\* GetDeleteCommand();  
 17 [VB] Public Function GetDeleteCommand() As SqlCommand  
 18 [JScript] public function GetDeleteCommand() : SqlCommand;

#### 20 *Description*

21 Gets the automatically generated Transact-SQL statement required to  
 22 perform deletions on the database when an application calls  
 23 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** on the  
 24 **System.Data.SqlClient.SqlDataAdapter** .

25 *Return Value:* The text of the Transact-SQL statement to be executed.

An application can use the **System.Data.SqlClient.SqlCommandBuilder.GetDeleteCommand** method for informational or troubleshooting purposes because it returns the text of the statement to be executed.

#### GetInsertCommand

```
[C#]      public      SqlCommand      GetInsertCommand();
[C++]     public:      SqlCommand*      GetInsertCommand();
[VB]     Public      Function      GetInsertCommand() As      SqlCommand
[JScript] public      function      GetInsertCommand()      :      SqlCommand;
```

#### *Description*

Gets the automatically generated Transact-SQL statement required to perform inserts on the database when an application calls **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** on the **System.Data.SqlClient.SqlDataAdapter**.

*Return Value:* The text of the Transact-SQL statement to be executed.

An application can use the **System.Data.SqlClient.SqlCommandBuilder.GetInsertCommand** method for informational or troubleshooting purposes because it returns the text of the statement to be executed.

#### GetUpdateCommand

```
[C#]      public      SqlCommand      GetUpdateCommand();
[C++]     public:      SqlCommand*      GetUpdateCommand();
```

```

1  [VB]    Public    Function    GetUpdateCommand()    As    SqlCommand
2  [JScript]    public    function    GetUpdateCommand()    :    SqlCommand;

```

#### Description

Gets the automatically generated Transact-SQL statement required to perform updates on the database when an application calls **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** on the **System.Data.SqlClient.SqlDataAdapter**.

*Return Value:* The text of the Transact-SQL statement to be executed.

An application can use the **System.Data.SqlClient.SqlCommandBuilder.GetUpdateCommand** method for informational or troubleshooting purposes because it returns the text of the statement to be executed.

#### RefreshSchema

```

16 [C#]          public          void          RefreshSchema();
17 [C++]         public:         void          RefreshSchema();
18 [VB]          Public          Sub          RefreshSchema()
19 [JScript]     public          function     RefreshSchema();

```

#### Description

Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.

An application should call **System.Data.SqlClient.SqlCommandBuilder.RefreshSchema** whenever the

1 SELECT statement associated with the  
2 **System.Data.SqlClient.SqlCommandBuilder** changes.

3 SqlConnection class (System.Data.SqlClient)

4 ToString

5  
6  
7 *Description*

8 Represents an open connection to a SQL Server database. This class cannot  
9 be inherited.

10 A **System.Data.SqlClient.SqlConnection** object represents a unique  
11 session to a SQL Server data source. In the case of a client/server database system,  
12 it is equivalent to a network connection to the server.

13 SqlConnection

14 *Example Syntax:*

15 ToString

16  
17 [C#] public SqlConnection();

18 [C++] public: SqlConnection();

19 [VB] Public Sub New()

20 [JScript] public function SqlConnection(); Initializes a new instance of the  
21 **System.Data.SqlClient.SqlConnection** class.

22  
23 *Description*

24 Initializes a new instance of the **System.Data.SqlClient.SqlConnection**  
25 class.

When a new instance of **System.Data.SqlClient.SqlConnection** is created, the read/write properties are set to the following initial values unless they are specifically set using their associated keywords in the **System.Data.SqlClient.SqlConnection.ConnectionString** property.

**SqlConnection**

*Example Syntax:*

**ToString**

```
[C#]      public      SqlConnection(string      connectionString);
[C++]      public:      SqlConnection(String*      connectionString);
[VB]      Public      Sub      New(ByVal      connectionString      As      String)
[JScript]      public      function      SqlConnection(connectionString      :      String);
```

#### *Description*

Initializes a new instance of the **System.Data.SqlClient.SqlConnection** class when given a string containing the connection string.

When a new instance of **System.Data.SqlClient.SqlConnection** is created, the read/write properties are set to the following initial values unless they are specifically set using their associated keywords in the **System.Data.SqlClient.SqlConnection.ConnectionString** property. The connection used to open the SQL Server database.

**ConnectionString**

**ToString**

```
[C#]      public      string      ConnectionString      {get;      set;}
```

```

1  [C++] public: __property String* get_ConnectionString();public: __property void
2  set_ConnectionString(String*);
3  [VB]      Public      Property      ConnectionString      As      String
4  [JScript] public function get ConnectionString() : String;public function set
5  ConnectionString(String);

```

### 7 *Description*

8 Gets or sets the string used to open a SQL Server database.

9 The **System.Data.SqlClient.SqlConnection.ConnectionString** is similar  
10 to an OLE DB connection string, but is not identical. Unlike OLE DB or ADO, the  
11 connection string that is returned is the same as the user-set  
12 **System.Data.SqlClient.SqlConnection.ConnectionString** minus security  
13 information if Persist Security Info value is set to **false** (default). The SQL Server  
14 .NET Data Provider does not persist or return the password in a connection string  
15 unless you set Persist Security Info to **true** .

16 ConnectionTimeout

17 ToString

```

18
19 [C#]      public      int      ConnectionTimeout      {get;}
20 [C++]      public:      __property      int      get_ConnectionTimeout();
21 [VB]      Public      ReadOnly      Property      ConnectionTimeout      As      Integer
22 [JScript]      public      function      get      ConnectionTimeout()      :      int;

```

### 24 *Description*

25

1 Gets the time to wait while trying to establish a connection before  
2 terminating the attempt and generating an error.

3 A value of 0 indicates no limit, and should be avoided in a  
4 **System.Data.SqlClient.SqlConnection.ConnectionString** because an attempt to  
5 connect will wait indefinitely.

6 Container

7 Database

8 ToString

9  
10  
11 *Description*

12 Gets the name of the current database or the database to be used once a  
13 connection is open.

14 The **System.Data.SqlClient.SqlConnection.Database** property updates  
15 dynamically. If you change the current database using a Transact-SQL statement  
16 or the **System.Data.SqlClient.SqlConnection.ChangeDatabase(System.String)**  
17 method, an informational message is sent and the property is updated  
18 automatically.

19 DataSource

20 ToString

21  
22 [C#] public string DataSource {get;}  
23 [C++] public: \_\_property String\* get\_DataSource();  
24 [VB] Public ReadOnly Property DataSource As String  
25 [JScript] public function get DataSource() : String;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Gets the name of the instance of SQL Server to which to connect.

DesignMode

Events

PacketSize

ToString

*Description*

Gets the size (in bytes) of network packets used to communicate with an instance of SQL Server .

If an application performs bulk copy operations, or sends or receives large amounts of **text** or **image** data, a packet size larger than the default may improve efficiency because it results in fewer network read and write operations. If an application sends and receives small amounts of information, you can set the packet size to 512 bytes (using the Packet Size value in the **System.Data.SqlClient.SqlConnection.ConnectionString** ), which is sufficient for most data transfer operations. For most applications, the default packet size is best.

ServerVersion

ToString

```
[C#]          public          string          ServerVersion          {get;}
[C++]         public:         __property      String*          get_ServerVersion();
```

```
1 [VB] Public ReadOnly Property ServerVersion As String
2 [JScript] public function get ServerVersion() : String;
```

#### 4 *Description*

5 Gets a string containing the version of the instance of SQL Server to which  
6 the client is connected.

7 The version is of the form **##.##.####**, where the first two digits are the  
8 major version, the next two digits are the minor version, and the last four digits are  
9 the release version. The string is of the form **major.minor.build**, where major and  
10 minor are exactly two digits and build is exactly four digits.

11 Site

12 State

13 ToString

#### 16 *Description*

17 Gets the current state of the connection.

18 The allowed state changes are: From **Closed** to **Open** , using the **Open**  
19 method of the connection object.

20 WorkstationId

21 ToString

```
23 [C#] public string WorkstationId {get;}
```

```
24 [C++] public: __property String* get_WorkstationId();
```

```
25 [VB] Public ReadOnly Property WorkstationId As String
```

1 [JScript] public function get WorkstationId() : String;

2

3 *Description*

4 Gets a string that identifies the database client.

5 The string typically contains the network name of the client. The  
6 **System.Data.SqlClient.SqlConnection.WorkstationId** property corresponds to  
7 the **Workstation ID** connection string property.

8 ToString

9

10

11 *Description*

12 Occurs when an informational message is added.

13 ToString

14

15 [C#] public event StateChangeEventHandler StateChange;

16 [C++] public: \_\_event StateChangeEventHandler\* StateChange;

17 [VB] Public Event StateChange As StateChangeEventHandler

18

19 *Description*

20 Occurs when the state of the connection changes.

21 The **System.Data.SqlClient.SqlConnection.StateChange** event fires  
22 whenever the **System.Data.SqlClient.SqlConnection.State** changes from closed  
23 to opened, or from opened to closed.

24 BeginTransaction

25

```

1
2 [C#]          public          SqlTransaction          BeginTransaction();
3 [C++]         public:         SqlTransaction*         BeginTransaction();
4 [VB]   Public   Function   BeginTransaction()   As   SqlTransaction
5 [JScript] public function BeginTransaction() : SqlTransaction; Begins a database
6 transaction.

```

#### 8 *Description*

9            Begins                            a                            database                            transaction.

10 *Return Value:* An object representing the new transaction.

11            This command maps to the SQL Server implementation of BEGIN  
12 TRANSACTION.

13            BeginTransaction

```

14
15 [C#]   public   SqlTransaction   BeginTransaction(IsolationLevel   iso);
16 [C++]  public:  SqlTransaction*  BeginTransaction(IsolationLevel   iso);
17 [VB]   Public   Function   BeginTransaction(ByVal iso As IsolationLevel) As
18 SqlTransaction
19 [JScript] public function BeginTransaction(iso : IsolationLevel) : SqlTransaction;
20

```

#### 21 *Description*

22            Begins a database transaction with the specified isolation level.

23 *Return Value:* An object representing the new transaction.

24            This command maps to the SQL Server implementation of BEGIN  
25 TRANSACTION. The isolation level under which the transaction should run.

## BeginTransaction

```

1      BeginTransaction
2
3  [C#]   public   SqlTransaction   BeginTransaction(string   transactionName);
4  [C++]  public:   SqlTransaction*   BeginTransaction(String*   transactionName);
5  [VB]   Public Function BeginTransaction(ByVal transactionName As String) As
6  SqlTransaction
7  [JScript] public function BeginTransaction(transactionName : String) :
8  SqlTransaction;           Begins           a           database           transaction.
9

```

### *Description*

Begins a database transaction with the specified transaction name.

*Return Value:* An object representing the new transaction.

This command maps to the SQL Server implementation of BEGIN TRANSACTION. The name of the transaction.

## BeginTransaction

```

17  [C#]   public   SqlTransaction   BeginTransaction(IsolationLevel   iso,   string
18  transactionName);
19  [C++]  public:   SqlTransaction*   BeginTransaction(IsolationLevel   iso,   String*
20  transactionName);
21  [VB]   Public Function BeginTransaction(ByVal iso As IsolationLevel, ByVal
22  transactionName           As           String)           As           SqlTransaction
23  [JScript] public function BeginTransaction(iso : IsolationLevel, transactionName :
24  String)           :           SqlTransaction;
25

```

## Description

Begins a database transaction with the specified isolation level and transaction name.

**Return Value:** An object representing the new transaction.

This command maps to the SQL Server implementation of BEGIN TRANSACTION. The isolation level under which the transaction should run. The name of the transaction.

## ChangeDatabase

[C#]        public        void        ChangeDatabase(string        database);

[C++]    public:    \_\_sealed    void    ChangeDatabase(String\*    database);

[VB] NotOverridable Public Sub ChangeDatabase(ByVal database As String)

[JScript]    public    function    ChangeDatabase(database    :    String);

## Description

Changes the current database for an open **System.Data.SqlClient.SqlConnection**.

The value supplied in the *database* parameter must be a valid database name. The *database* parameter cannot contain a null value, be empty, or contain a string with only blank characters. The database name.

## Close

[C#]        public        void        Close();

[C++]        public:        \_\_sealed        void        Close();

1	[VB]	NotOverridable	Public	Sub	Close()
2	[JScript]	public		function	Close();

3

4 *Description*

5 Closes the connection to the database. This is the preferred method of

6 closing any open connection.

7 The **System.Data.SqlClient.SqlConnection.Close** method rolls back any

8 pending transactions. It then releases the connection to the connection pool, or

9 closes the connection if connection pooling is disabled.

10 CreateCommand

11					
12	[C#]	public	SqlCommand		CreateCommand();
13	[C++]	public:	SqlCommand*		CreateCommand();
14	[VB]	Public	Function	CreateCommand()	As SqlCommand
15	[JScript]	public	function	CreateCommand()	: SqlCommand;

16

17 *Description*

18 Creates and returns a **System.Data.SqlClient.SqlCommand** object

19 associated with the **System.Data.SqlClient.SqlConnection**.

20 *Return Value:* A **System.Data.SqlClient.SqlCommand** object.

21 Dispose

22					
23	[C#]	protected	override	void	Dispose(bool disposing);
24	[C++]	protected:		void	Dispose(bool disposing);
25	[VB]	Overrides	Protected	Sub	Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean); Releases the resources used by the **System.Data.SqlClient.SqlConnection** .

#### *Description*

Releases the unmanaged resources used by the **System.Data.SqlClient.SqlConnection** and optionally releases the managed resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

#### Open

[C#]	public	void	Open();
[C++]	public:	__sealed	void Open();
[VB]	NotOverridable	Public	Sub Open()
[JScript]	public	function	Open();

#### *Description*

Opens a database connection with the property settings specified by the **System.Data.SqlClient.SqlConnection.ConnectionString** .

The **System.Data.SqlClient.SqlConnection** draws an open connection from the connection pool if one is available. Otherwise, it establishes a new connection to an instance of SQL Server.

#### IdbConnection.BeginTransaction

```

1
2 [C#]          IDbTransaction          IDbConnection.BeginTransaction();
3 [C++]          IDbTransaction*          IDbConnection::BeginTransaction();
4 [VB]  Function  BeginTransaction()  As  IDbTransaction  Implements
5 IDbConnection.BeginTransaction
6 [JScript] function IDbConnection.BeginTransaction() : IDbTransaction;
7         IDbConnection.BeginTransaction
8
9 [C#]  IDbTransaction  IDbConnection.BeginTransaction(IsolationLevel  iso);
10 [C++] IDbTransaction* IDbConnection::BeginTransaction(IsolationLevel  iso);
11 [VB] Function BeginTransaction(ByVal iso As IsolationLevel) As IDbTransaction
12 Implements                                     IDbConnection.BeginTransaction
13 [JScript] function  IDbConnection.BeginTransaction(iso : IsolationLevel) :
14 IDbTransaction;
15         IDbConnection.CreateCommand
16
17 [C#]          IDbCommand          IDbConnection.CreateCommand();
18 [C++]          IDbCommand*          IDbConnection::CreateCommand();
19 [VB]  Function  CreateCommand()  As  IDbCommand  Implements
20 IDbConnection.CreateCommand
21 [JScript] function IDbConnection.CreateCommand() : IDbCommand;
22         ICloneable.Clone
23
24 [C#]          object          ICloneable.Clone();
25 [C++]          Object*          ICloneable::Clone();

```

1 [VB] Function Clone() As Object Implements ICloneable.Clone

2 [JScript] function ICloneable.Clone() : Object;

3 SqlDataAdapter class (System.Data.SqlClient)

4 ToString

5

6

7 *Description*

8 Represents a set of data commands and a database connection which are  
9 used to fill the **System.Data.DataSet** and update a SQL Server database. This  
10 class cannot be inherited.

11 The **System.Data.SqlClient.SqlDataAdapter** , serves as a bridge between  
12 a **System.Data.DataSet** and SQL Server for retrieving and saving data. The  
13 **System.Data.SqlClient.SqlDataAdapter** provides this bridge by mapping  
14 **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** , which  
15 changes the data in the **System.Data.DataSet** to match the data in the data source,  
16 and **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** ,  
17 which changes the data in the data source to match the data in the  
18 **System.Data.DataSet** , using the appropriate Transact-SQL statements against the  
19 data source.

20 SqlDataAdapter

21 *Example Syntax:*

22 ToString

23

24 [C#] public SqlDataAdapter();

25 [C++] public: SqlDataAdapter();

1        [VB]                                Public                                Sub                                New()  
 2        [JScript] public function SqlDataAdapter(); Initializes a new instance of the  
 3        **System.Data.SqlClient.SqlDataAdapter**                                class.

4  
 5        *Description*

6                Initializes a new instance of the **System.Data.SqlClient.SqlDataAdapter**  
 7        class.

8                When an instance of **System.Data.SqlClient.SqlDataAdapter** is created,  
 9        the following read/write properties are set to the following initial values.

10                SqlDataAdapter

11                *Example Syntax:*

12                ToString

13  
 14        [C#]        public        SqlDataAdapter(SqlCommand        selectCommand);

15        [C++]        public:        SqlDataAdapter(SqlCommand\*        selectCommand);

16        [VB]        Public        Sub        New(ByVal        selectCommand        As        SqlCommand)

17        [JScript]        public        function        SqlDataAdapter(selectCommand :        SqlCommand);

18  
 19        *Description*

20                Initializes a new instance of the **System.Data.SqlClient.SqlDataAdapter**  
 21        class with the specified Transact-SQL SELECT statement.

22                When an instance of **System.Data.SqlClient.SqlDataAdapter** is created,  
 23        the following read/write properties are set to the following initial values. A  
 24        **System.Data.SqlClient.SqlCommand** that is a Transact-SQL SELECT  
 25        statement.

SqlDataAdapter

*Example Syntax:*

ToString

```
[C#] public SqlDataAdapter(string selectCommandText, SqlConnection
selectConnection);
```

```
[C++] public: SqlDataAdapter(String* selectCommandText, SqlConnection*
selectConnection);
```

```
[VB] Public Sub New(ByVal selectCommandText As String, ByVal
selectConnection As SqlConnection)
```

```
[JScript] public function SqlDataAdapter(selectCommandText : String,
selectConnection : SqlConnection);
```

#### *Description*

Initializes a new instance of the **System.Data.SqlClient.SqlDataAdapter** class with a **System.Data.SqlClient.SqlDataAdapter.SelectCommand** and a **System.Data.SqlClient.SqlConnection** object.

This implementation of the **System.Data.SqlClient.SqlDataAdapter** opens and closes a **System.Data.SqlClient.SqlConnection** if it is not already open. This can be useful in a an application that must call the **System.Data.Common.DbDataAdapter.Fill(System.Data.DataTable)** method for two or more **System.Data.SqlClient.SqlDataAdapter** objects. If the **System.Data.SqlClient.SqlConnection** is already open, you must explicitly call **System.Data.SqlClient.SqlConnection.Close** or **System.Data.SqlClient.SqlConnection.Dispose(System.Boolean)** to close it.

The **System.Data.SqlClient.SqlDataAdapter.SelectCommand** . A  
**System.Data.SqlClient.SqlConnection** that represents the connection.

**SqlDataAdapter**

*Example Syntax:*

**ToString**

[C#] public SqlDataAdapter(string selectCommandText, string  
 selectConnectionString);

[C++] public: SqlDataAdapter(String\* selectCommandText, String\*  
 selectConnectionString);

[VB] Public Sub New(ByVal selectCommandText As String, ByVal  
 selectConnectionString As String)

[JScript] public function SqlDataAdapter(selectCommandText : String,  
 selectConnectionString : String);

### *Description*

Initializes a new instance of the **System.Data.SqlClient.SqlDataAdapter**  
 class with a **System.Data.SqlClient.SqlDataAdapter.SelectCommand** and a  
 connection string.

When an instance of **System.Data.SqlClient.SqlDataAdapter** is created,  
 the following read/write properties are set to the following initial values. The  
**System.Data.SqlClient.SqlDataAdapter.SelectCommand** . The connection  
 string.

**AcceptChangesDuringFill**

**Container**

1 DeleteCommand

2 ToString

3

4

5 *Description*

6 Gets or sets a Transact-SQL statement or stored procedure to delete records  
7 from the data set.

8 During

9 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** , if this  
10 property is not set and primary key information is present in the  
11 **System.Data.DataSet** , the  
12 **System.Data.SqlClient.SqlDataAdapter.DeleteCommand** can be generated  
13 automatically if you set the  
14 **System.Data.OleDb.OleDbDataAdapter.SelectCommand** property and use the  
15 **System.Data.SqlClient.SqlCommandBuilder** . Then, any additional commands  
16 that you do not set are generated by the  
17 **System.Data.SqlClient.SqlCommandBuilder** . This generation logic requires  
18 key column information to be present in the **System.Data.DataSet** . For more  
19 information see .

20 DesignMode

21 Events

22 InsertCommand

23 ToString

24

25

1  
2  
3 *Description*

4 Gets or sets a Transact-SQL statement or stored procedure to insert new  
5 records into the data source.

6 During

7 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** , if this  
8 property is not set and primary key information is present in the  
9 **System.Data.DataSet** , the  
10 **System.Data.SqlClient.SqlDataAdapter.InsertCommand** can be generated  
11 automatically if you set the  
12 **System.Data.OleDb.OleDbDataAdapter.SelectCommand** property and use the  
13 **System.Data.SqlClient.SqlCommandBuilder** . Then, any additional commands  
14 that you do not set are generated by the  
15 **System.Data.SqlClient.SqlCommandBuilder** . This generation logic requires  
16 key column information to be present in the **System.Data.DataSet** . For more  
17 information see .

18 MissingMappingAction

19 MissingSchemaAction

20 SelectCommand

21 ToString

22  
23  
24 *Description*  
25

Gets or sets a Transact-SQL statement or stored procedure used to select records in the data source.

When **System.Data.SqlClient.SqlDataAdapter.SelectCommand** is assigned to a previously created **System.Data.SqlClient.SqlCommand**, the **System.Data.SqlClient.SqlCommand** is not cloned. The **System.Data.SqlClient.SqlDataAdapter.SelectCommand** maintains a reference to the previously created **System.Data.SqlClient.SqlCommand** object.

Site

TableMappings

UpdateCommand

ToString

### *Description*

Gets or sets a Transact-SQL statement or stored procedure used to update records in the data source.

During

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**, if this property is not set and primary key information is present in the **System.Data.DataSet**, the **System.Data.SqlClient.SqlDataAdapter.UpdateCommand** can be generated automatically if you set the **System.Data.OleDb.OleDbDataAdapter.SelectCommand** property and use the **System.Data.SqlClient.SqlCommandBuilder**. Then, any additional commands that you do not set are generated by the

**System.Data.SqlClient.SqlCommandBuilder** . This generation logic requires key column information to be present in the **System.Data.DataSet** . For more information see .

ToString

### Description

Occurs during **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** after a command is executed against the data source. The attempt to update is made, so the event fires.

When using **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** , there are two events that occur per data row updated. The order of execution is as follows: The values in the **System.Data.DataRow** are moved to the parameter values.

ToString

```
[C#]    public    event    SqlRowUpdatingEventHandler    RowUpdating;
[C++]   public:    __event    SqlRowUpdatingEventHandler*    RowUpdating;
[VB]    Public    Event    RowUpdating    As    SqlRowUpdatingEventHandler
```

### Description

Occurs during **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** before a

command is executed against the data source. The attempt to update is made, so the event fires.

When using **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**, there are two events that occur per data row updated. The order of execution is as follows: The values in the **System.Data.DataRow** are moved to the parameter values.

#### CreateRowUpdatedEvent

[C#]                   protected                   override                   RowUpdatedEventArgs

CreateRowUpdatedEvent(DataRow   dataRow,   IDbCommand   command,

StatementType   statementType,   DataTableMapping   tableMapping);

[C++] protected: RowUpdatedEventArgs\* CreateRowUpdatedEvent(DataRow\*

dataRow,   IDbCommand\*   command,   StatementType   statementType,

DataTableMapping\*                   tableMapping);

[VB] Overrides Protected Function CreateRowUpdatedEvent(ByVal dataRow As

DataRow, ByVal command As IDbCommand, ByVal statementType As

StatementType, ByVal tableMapping As DataTableMapping) As

RowUpdatedEventArgs

[JScript] protected override function CreateRowUpdatedEvent(dataRow :

DataRow, command : IDbCommand, statementType : StatementType,

tableMapping : DataTableMapping) : RowUpdatedEventArgs;

#### *Description*

#### CreateRowUpdatingEvent

```

1
2 [C#]          protected          override          RowUpdatingEventArgs
3 CreateRowUpdatingEvent(DataRow  dataRow,  IDbCommand  command,
4 StatementType  statementType,  DataTableMapping  tableMapping);
5 [C++] protected: RowUpdatingEventArgs* CreateRowUpdatingEvent(DataRow*
6 dataRow,  IDbCommand*  command,  StatementType  statementType,
7 DataTableMapping*      tableMapping);
8 [VB] Overrides Protected Function CreateRowUpdatingEvent(ByVal dataRow As
9 DataRow, ByVal  command As IDbCommand, ByVal  statementType As
10 StatementType, ByVal  tableMapping As  DataTableMapping) As
11 RowUpdatingEventArgs
12 [JScript] protected override function CreateRowUpdatingEvent(dataRow :
13 DataRow, command : IDbCommand, statementType : StatementType,
14 tableMapping :  DataTableMapping) :  RowUpdatingEventArgs;

```

### *Description*

The following example fills a **System.Data.DataSet** with the schema only, while filling a **System.Data.DataTable** with records, when provided a source table.

### *Dispose*

```

22 [C#]          protected          override          void          Dispose(bool  disposing);
23 [C++]          protected:          void          Dispose(bool  disposing);
24 [VB] Overrides Protected Sub  Dispose(ByVal  disposing As Boolean)
25 [JScript] protected override function Dispose(disposing : Boolean); Releases the

```

resources used by the **System.Data.SqlClient.SqlDataAdapter** .

### *Description*

Releases the unmanaged resources used by the **System.Data.SqlClient.SqlDataAdapter** and optionally releases the managed resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

### **OnRowUpdated**

[C#] protected override void OnRowUpdated(RowUpdatedEventArgs value);

[C++] protected: void OnRowUpdated(RowUpdatedEventArgs\* value);

[VB] Overrides Protected Sub OnRowUpdated(ByVal value As RowUpdatedEventArgs)

[JScript] protected override function OnRowUpdated(value : RowUpdatedEventArgs);

### *Description*

Raises the **System.Data.SqlClient.SqlDataAdapter.RowUpdated** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Data.SqlClient.SqlRowUpdatedEventArgs** that contains the event data.

### **OnRowUpdating**

```

1
2 [C#] protected override void OnRowUpdating(RowUpdatingEventArgs value);
3 [C++] protected: void OnRowUpdating(RowUpdatingEventArgs* value);
4 [VB] Overrides Protected Sub OnRowUpdating(ByVal value As
5 RowUpdatingEventArgs)
6 [JScript] protected override function OnRowUpdating(value :
7 RowUpdatingEventArgs);
8

```

#### 9 *Description*

10 Raises the **System.Data.SqlClient.SqlDataAdapter.RowUpdating** event.

11 Raising an event invokes the event handler through a delegate. For more  
12 information, see . A **System.Data.SqlClient.SqlRowUpdatingEventArgs** that  
13 contains the event data.

14 **ICloneable.Clone**

```

15
16 [C#] object ICloneable.Clone();
17 [C++] Object* ICloneable::Clone();
18 [VB] Function Clone() As Object Implements ICloneable.Clone
19 [JScript] function ICloneable.Clone() : Object;

```

20 **SqlDataReader class (System.Data.SqlClient)**

21 **Update**

#### 24 *Description*

25

Provides a means of reading a forward-only stream of rows from a SQL Server database. This class cannot be inherited.

To create a **System.Data.SqlClient.SqlDataReader** , you must call the **System.Data.SqlClient.SqlCommand.ExecuteReader** method of the **System.Data.SqlClient.SqlCommand** object, rather than directly using a constructor.

Depth

Update

[C#]	public	int	Depth	{get;}
[C++]	public:	__property	int	get_Depth();
[VB]	Public	ReadOnly	Property	Depth As Integer
[JScript]	public	function	get	Depth() : int;

### Description

Gets a value indicating the depth of nesting for the current row.

The outermost table has a depth of zero. The SQL Server .NET Data Provider does not support nesting and always returns zero.

FieldCount

Update

[C#]	public	int	FieldCount	{get;}
[C++]	public:	__property	int	get_FieldCount();
[VB]	Public	ReadOnly	Property	FieldCount As Integer
[JScript]	public	function	get	FieldCount() : int;

## Description

Gets the number of columns in the current row.

After executing a query that does not return rows (for example, using the **System.Data.SqlClient.SqlCommand.ExecuteNonQuery** method), **System.Data.SqlClient.SqlDataReader.FieldCount** returns -1.

IsClosed

Update

[C#]	public	bool	IsClosed	{get;}
[C++]	public:	__property	bool	get_IsClosed();
[VB]	Public	ReadOnly	Property	IsClosed As Boolean
[JScript]	public	function	get	IsClosed() : Boolean;

## Description

Gets a value indicating whether the data reader is closed.

**System.Data.SqlClient.SqlDataReader.IsClosed** and **System.Data.SqlClient.SqlDataReader.RecordsAffected** are the only properties that you can call after the **System.Data.SqlClient.SqlDataReader** is closed.

Item

Update

[C#]	public	object	this[string	name]	{get;}
[C++]	public:	__property	Object*	get_Item(String*	name);
[VB]	Public	Default	ReadOnly	Property	Item(ByVal name As String) As Object

1 [JScript]        returnValue        =        SqlDataReaderObject.Item(name);

2

3 *Description*

4        Gets the value of the specified column in its native format given the column  
5 name. The column name.

6        Item

7        Update

8

9 [C#]        public        object        this[int        i]        {get;}

10 [C++]        public:        \_\_property        Object\*        get\_Item(int        i);

11 [VB] Public Default ReadOnly Property Item(ByVal i As Integer) As Object

12 [JScript] returnValue = SqlDataReaderObject.Item(i); Gets the value of a column  
13 in                    its                    native                    format.

14

15 *Description*

16        Gets the value of the specified column in its native format given the column  
17 ordinal. The zero-based column ordinal.

18        RecordsAffected

19        Update

20

21 [C#]        public        int        RecordsAffected        {get;}

22 [C++]        public:        \_\_property        int        get\_RecordsAffected();

23 [VB] Public ReadOnly Property RecordsAffected As Integer

24 [JScript]        public        function        get        RecordsAffected()        :        int;

25

## Description

Gets the number of rows changed, inserted, or deleted by execution of the Transact-SQL statement.

The **System.Data.SqlClient.SqlDataReader.RecordsAffected** property is not set until all rows are read and you close the **System.Data.SqlClient.SqlDataReader**.

## Close

[C#]	public	void	Close();
[C++]	public:      __sealed	void	Close();
[VB]	NotOverridable	Public Sub	Close()
[JScript]	public	function	Close();

## Description

Closes the **System.Data.SqlClient.SqlDataReader** object.

You must explicitly call the **System.Data.SqlClient.SqlDataReader.Close** method when you are through using the **System.Data.SqlClient.SqlDataReader** to use the associated **System.Data.SqlClient.SqlConnection** for any other purpose.

## GetBoolean

[C#]	public	bool	GetBoolean(int i);
[C++]	public:      __sealed	bool	GetBoolean(int i);
[VB]	NotOverridable Public Function	GetBoolean(ByVal i As Integer) As	

Boolean

[JScript] public function GetBoolean(i : int) : Boolean;

*Description*

Gets the value of the specified column as a boolean.

*Return Value:* The value of the column.

No conversions are performed, therefore the data retrieved must already be a boolean or an exception is generated. The zero-based column ordinal.

GetByte

[C#] public byte GetByte(int i);

[C++] public: \_\_sealed unsigned char GetByte(int i);

[VB] NotOverridable Public Function GetByte(ByVal i As Integer) As Byte

[JScript] public function GetByte(i : int) : Byte;

*Description*

Gets the value of the specified column as a byte.

*Return Value:* The value of the specified column as a byte.

No conversions are performed, therefore the data retrieved must already be a byte. The zero-based column ordinal.

GetBytes

[C#] public long GetBytes(int i, long dataIndex, byte[] buffer, int bufferIndex, int length);

[C++] public: \_\_sealed \_\_int64 GetBytes(int i, \_\_int64 dataIndex, unsigned char

```

1  buffer    __gc[],          int    bufferIndex,          int    length);
2  [VB] NotOverridable Public Function GetBytes(ByVal i As Integer, ByVal
3  dataIndex As Long, ByVal buffer() As Byte, ByVal bufferIndex As Integer,
4  ByVal     length          As     Integer)          As     Long
5  [JScript] public function GetBytes(i : int, dataIndex : long, buffer : Byte[],
6  bufferIndex : int, length : int) : long;

```

### Description

Reads a stream of bytes from the specified column offset into the buffer an array starting at the given buffer offset.

*Return Value:* The actual number of bytes read.

The actual number of bytes read can be less than the requested length, if the end of the row is reached. If you pass a buffer that is **null** , **System.Data.SqlClient.SqlDataReader.GetBytes(System.Int32, System.Int64, System.Byte[], System.Int32, System.Int32)** returns the length of the row in bytes. The zero-based column ordinal. The index within the field from which to begin the read operation. The buffer into which to read the stream of bytes. The index for *buffer* to begin the read operation. The maximum length to copy into the buffer.

### GetChar

```

21 [C#]          public          char          GetChar(int          i);
22 [C++]          public:          __sealed          __wchar_t          GetChar(int          i);
23 [VB] NotOverridable Public Function GetChar(ByVal i As Integer) As Char
24 [JScript] public function GetChar(i : int) : Char;

```

1  
2 *Description*

3 Gets the value of the specified column as a single character.

4 *Return Value:* The value of the specified column.

5 No conversions are performed, therefore the data retrieved must already be  
6 a character. The zero-based column ordinal.

7 **GetChars**

8  
9 [C#] public long GetChars(int i, long dataIndex, char[] buffer, int bufferIndex, int  
10 length);

11 [C++] public: \_\_sealed \_\_int64 GetChars(int i, \_\_int64 dataIndex, \_\_wchar\_t  
12 buffer \_\_gc[], int bufferIndex, int length);

13 [VB] NotOverridable Public Function GetChars(ByVal i As Integer, ByVal  
14 dataIndex As Long, ByVal buffer() As Char, ByVal bufferIndex As Integer,  
15 ByVal length As Integer) As Long

16 [JScript] public function GetChars(i : int, dataIndex : long, buffer : Char[],  
17 bufferIndex : int, length : int) : long;

18  
19 *Description*

20 Reads a stream of characters from the specified column offset into the  
21 buffer as an array starting at the given buffer offset.

22 *Return Value:* The actual number of characters read.

23 The actual number of characters read can be less than the requested length,  
24 if the end of the field is reached. If you pass a buffer that is **null** ,  
25 **System.Data.SqlClient.SqlDataReader.GetChars(System.Int32, System.Int64,**

**System.Char[],System.Int32,System.Int32)** returns the length of the field in characters. The zero-based column ordinal. The index within the row from which to begin the read operation. The buffer into which to copy data. The index for *buffer* to begin the read operation. The number of characters to read.

#### GetData

```
[C#]          public          IDataReader          GetData(int          i);
[C++]         public:         __sealed          IDataReader*          GetData(int          i);
[VB] NotOverridable Public Function GetData(ByVal i As Integer) As
IDataReader
[JScript]     public  function  GetData(i      :  int)      :  IDataReader;
```

#### Description

Not currently supported. The zero-based column ordinal.

#### GetDataTypeName

```
[C#]          public          string          GetDataTypeName(int          i);
[C++]         public:         __sealed          String*          GetDataTypeName(int          i);
[VB] NotOverridable Public Function GetDataTypeName(ByVal i As Integer) As
String
[JScript]     public  function  GetDataTypeName(i      :  int)      :  String;
```

#### Description

1 Gets the name of the source data type.  
 2 *Return Value:* The name of the back-end data type. The zero-based column  
 3 ordinal.

4 **GetDateTime**

5  
 6 [C#] public DateTime GetDateTime(int i);  
 7 [C++] public: \_\_sealed DateTime GetDateTime(int i);  
 8 [VB] NotOverridable Public Function GetDateTime(ByVal i As Integer) As  
 9 DateTime  
 10 [JScript] public function GetDateTime(i : int) : DateTime;

11  
 12 *Description*

13 Gets the value of the specified column as a **System.DateTime** object.  
 14 *Return Value:* The value of the specified column.

15 No conversions are performed, therefore the data retrieved must already be  
 16 a **System.DateTime** object. The zero-based column ordinal.

17 **GetDecimal**

18  
 19 [C#] public decimal GetDecimal(int i);  
 20 [C++] public: \_\_sealed Decimal GetDecimal(int i);  
 21 [VB] NotOverridable Public Function GetDecimal(ByVal i As Integer) As  
 22 Decimal  
 23 [JScript] public function GetDecimal(i : int) : Decimal;

24  
 25 *Description*

1 Gets the value of the specified column as a **System.Decimal** object.

2 *Return Value:* The value of the specified column.

3 No conversions are performed, therefore the data retrieved must already be  
4 a **System.Decimal** object. The zero-based column ordinal.

5 **GetDouble**

6  
7 [C#] public double GetDouble(int i);

8 [C++] public: \_\_sealed double GetDouble(int i);

9 [VB] NotOverridable Public Function GetDouble(ByVal i As Integer) As Double

10 [JScript] public function GetDouble(i : int) : double;

11  
12 *Description*

13 Gets the value of the specified column as a double-precision floating point  
14 number.

15 *Return Value:* The value of the specified column.

16 No conversions are performed, therefore the data retrieved must already be  
17 a double-precision floating point number. The zero-based column ordinal.

18 **GetFieldType**

19  
20 [C#] public Type GetFieldType(int i);

21 [C++] public: \_\_sealed Type\* GetFieldType(int i);

22 [VB] NotOverridable Public Function GetFieldType(ByVal i As Integer) As Type

23 [JScript] public function GetFieldType(i : int) : Type;

24  
25 *Description*

Gets the **System.Type** that is the data type of the object.  
*Return Value:* The **System.Type** that is the data type of the object. The zero-based column ordinal.

#### GetFloat

```
[C#]          public          float          GetFloat(int          i);
[C++]         public:         __sealed        float          GetFloat(int          i);
[VB] NotOverridable Public Function GetFloat(ByVal i As Integer) As Single
[JScript]     public         function         GetFloat(i          :          int)          :          float;
```

#### Description

Gets the value of the specified column as a single-precision floating point number.

*Return Value:* The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a single-precision floating point number. The zero-based column ordinal.

#### GetGuid

```
[C#]          public          Guid          GetGuid(int          i);
[C++]         public:         __sealed        Guid          GetGuid(int          i);
[VB] NotOverridable Public Function GetGuid(ByVal i As Integer) As Guid
[JScript]     public         function         GetGuid(i          :          int)          :          Guid;
```

#### Description



1 Gets the value of the specified column as a 32-bit signed integer.

2 *Return Value:* The value of the specified column.

3 No conversions are performed, therefore the data retrieved must already be  
4 a 32-bit signed integer. The zero-based column ordinal.

5 GetInt64

6  
7 [C#] public long GetInt64(int i);

8 [C++] public: \_\_sealed \_\_int64 GetInt64(int i);

9 [VB] NotOverridable Public Function GetInt64(ByVal i As Integer) As Long

10 [JScript] public function GetInt64(i : int) : long;

11  
12 *Description*

13 Gets the value of the specified column as a 64-bit signed integer.

14 *Return Value:* The value of the specified column.

15 No conversions are performed, therefore the data retrieved must already be  
16 a 64-bit signed integer. The zero-based column ordinal.

17 GetName

18  
19 [C#] public string GetName(int i);

20 [C++] public: \_\_sealed String\* GetName(int i);

21 [VB] NotOverridable Public Function GetName(ByVal i As Integer) As String

22 [JScript] public function GetName(i : int) : String;

23  
24 *Description*

25



For the **System.Data.SqlClient.SqlDataReader.GetSchemaTable** method returns metadata about each column in the following order: DataReader Column Description ColumnName The name of the column; this might not be unique. If this cannot be determined, a null value is returned. This name always reflects the most recent renaming of the column in the current view or command text.

#### GetSqlBinary

```
[C#]      public      SqlBinary      GetSqlBinary(int      i);
[C++]     public:      SqlBinary      GetSqlBinary(int      i);
[VB]      Public Function GetSqlBinary(ByVal i As Integer) As SqlBinary
[JScript] public function GetSqlBinary(i : int) : SqlBinary;
```

#### *Description*

Gets the value of the specified column as a **System.Data.SqlTypes.SqlBinary**.

*Return Value:* A **System.Data.SqlTypes.SqlBinary**. The zero-based column ordinal.

#### GetSqlBoolean

```
[C#]      public      SqlBoolean      GetSqlBoolean(int      i);
[C++]     public:      SqlBoolean      GetSqlBoolean(int      i);
[VB]      Public Function GetSqlBoolean(ByVal i As Integer) As SqlBoolean
[JScript] public function GetSqlBoolean(i : int) : SqlBoolean;
```

#### GetSqlByte

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      SqlByte      GetSqlByte(int      i);
[C++]      public:      SqlByte      GetSqlByte(int      i);
[VB] Public Function GetSqlByte(ByVal i As Integer) As SqlByte
[JScript] public function GetSqlByte(i : int) : SqlByte;
```

*Description*

Gets the value of the specified column as a  
**System.Data.SqlTypes.SqlByte**.  
*Return Value:* A **System.Data.SqlTypes.SqlByte**. The zero-based column ordinal.

GetSqlDateTime

```
[C#]      public      SqlDateTime      GetSqlDateTime(int      i);
[C++]      public:      SqlDateTime      GetSqlDateTime(int      i);
[VB] Public Function GetSqlDateTime(ByVal i As Integer) As SqlDateTime
[JScript] public function GetSqlDateTime(i : int) : SqlDateTime;
```

*Description*

Gets the value of the specified column as a  
**System.Data.SqlTypes.SqlDateTime**.  
*Return Value:* A **System.Data.SqlTypes.SqlDateTime**. The zero-based column ordinal.

GetSqlDecimal

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      SqlDecimal      GetSqlDecimal(int      i);
[C++]      public:      SqlDecimal      GetSqlDecimal(int      i);
[VB] Public Function GetSqlDecimal(ByVal i As Integer) As SqlDecimal
[JScript] public function GetSqlDecimal(i : int) : SqlDecimal;
```

*Description*

Gets the value of the specified column as a  
**System.Data.SqlTypes.SqlDecimal** .  
*Return Value:* A **System.Data.SqlTypes.SqlDecimal** . The zero-based column ordinal.

GetSqlDouble

```
[C#]      public      SqlDouble      GetSqlDouble(int      i);
[C++]      public:      SqlDouble      GetSqlDouble(int      i);
[VB] Public Function GetSqlDouble(ByVal i As Integer) As SqlDouble
[JScript] public function GetSqlDouble(i : int) : SqlDouble;
```

*Description*

Gets the value of the specified column as a  
**System.Data.SqlTypes.SqlDouble** .  
*Return Value:* A **System.Data.SqlTypes.SqlDouble** . The zero-based column ordinal.

GetSqlGuid

```

1
2 [C#]      public      SqlGuid      GetSqlGuid(int      i);
3 [C++]     public:     SqlGuid      GetSqlGuid(int      i);
4 [VB]  Public Function GetSqlGuid(ByVal i As Integer) As SqlGuid
5 [JScript] public function GetSqlGuid(i : int) : SqlGuid;

```

#### 6 *Description*

7 Gets the value of the specified column as a  
8 **System.Data.SqlTypes.SqlGuid**

9 *Return Value:* A **System.Data.SqlTypes.SqlGuid** . The zero-based column  
10 ordinal.  
11

#### 12 **GetSqlInt16**

```

13
14 [C#]      public      SqlInt16     GetSqlInt16(int      i);
15 [C++]     public:     SqlInt16     GetSqlInt16(int      i);
16 [VB]  Public Function GetSqlInt16(ByVal i As Integer) As SqlInt16
17 [JScript] public function GetSqlInt16(i : int) : SqlInt16;

```

#### 18 *Description*

19 Gets the value of the specified column as a  
20 **System.Data.SqlTypes.SqlInt16**

21 *Return Value:* A **System.Data.SqlTypes.SqlInt16** . The zero-based column  
22 ordinal.  
23

#### 24 **GetSqlInt32**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      SqlInt32      GetSqlInt32(int      i);
[C++]      public:      SqlInt32      GetSqlInt32(int      i);
[VB] Public Function GetSqlInt32(ByVal i As Integer) As SqlInt32
[JScript] public function GetSqlInt32(i : int) : SqlInt32;
```

*Description*

Gets the value of the specified column as a **System.Data.SqlTypes.SqlInt32**.  
*Return Value:* A **System.Data.SqlTypes.SqlInt32**. The zero-based column ordinal.

GetSqlInt64

```
[C#]      public      SqlInt64      GetSqlInt64(int      i);
[C++]      public:      SqlInt64      GetSqlInt64(int      i);
[VB] Public Function GetSqlInt64(ByVal i As Integer) As SqlInt64
[JScript] public function GetSqlInt64(i : int) : SqlInt64;
```

*Description*

Gets the value of the specified column as a **System.Data.SqlTypes.SqlInt64**.  
*Return Value:* A **System.Data.SqlTypes.SqlInt64**. The zero-based column ordinal.

GetSqlMoney

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      SqlMoney      GetSqlMoney(int      i);
[C++]      public:      SqlMoney      GetSqlMoney(int      i);
[VB] Public Function GetSqlMoney(ByVal i As Integer) As SqlMoney
[JScript] public function GetSqlMoney(i : int) : SqlMoney;
```

*Description*

Gets the value of the specified column as a **System.Data.SqlTypes.SqlMoney**.

*Return Value:* A **System.Data.SqlTypes.SqlMoney**. The zero-based column ordinal.

**GetSqlSingle**

```
[C#]      public      SqlSingle      GetSqlSingle(int      i);
[C++]      public:      SqlSingle      GetSqlSingle(int      i);
[VB] Public Function GetSqlSingle(ByVal i As Integer) As SqlSingle
[JScript] public function GetSqlSingle(i : int) : SqlSingle;
```

*Description*

Gets the value of the specified column as a **System.Data.SqlTypes.SqlSingle**.

*Return Value:* A **System.Data.SqlTypes.SqlSingle**. The zero-based column ordinal.

**GetSqlString**

```

1
2 [C#]      public      SqlString      GetSqlString(int      i);
3 [C++]     public:     SqlString      GetSqlString(int      i);
4 [VB]      Public Function GetSqlString(ByVal i As Integer) As SqlString
5 [JScript] public      function      GetSqlString(i      :      int)      :      SqlString;
6

```

#### 7 *Description*

8 Gets the value of the specified column as a  
9 **System.Data.SqlTypes.SqlString** .

10 *Return Value:* A **System.Data.SqlTypes.SqlString** . The zero-based column  
11 ordinal.

#### 12 **GetSqlValue**

```

13
14 [C#]      public      object      GetSqlValue(int      i);
15 [C++]     public:     Object*      GetSqlValue(int      i);
16 [VB]      Public Function GetSqlValue(ByVal i As Integer) As Object
17 [JScript] public      function      GetSqlValue(i      :      int)      :      Object;
18

```

#### 19 *Description*

20 Gets an **System.Object** that is a representation of the underlying  
21 **System.Data.SqlDbTypeVariant** .

22 *Return Value:* An **System.Object** that is a representation of the underlying  
23 **System.Data.SqlDbTypeVariant** .

#### 24 **System.Data.SqlClient.SqlDataReader.GetSqlValue(System.Int32)**

25 returns data using the native SQL Server types. To retrieve data using the .Net

Framework types, see

**System.Data.SqlClient.SqlDataReader.GetValue(System.Int32)** . The zero-based column ordinal.

#### GetSqlValues

[C#] public int GetSqlValues(object[] values);

[C++] public: int GetSqlValues(Object\* values \_\_gc[]);

[VB] Public Function GetSqlValues(ByVal values() As Object) As Integer

[JScript] public function GetSqlValues(values : Object[]) : int;

#### Description

Gets all the attribute columns in the current row.

**Return Value:** The number of instances of **System.Object** in the array.

For most applications, the **System.Data.SqlClient.SqlDataReader.GetValues(System.Object[])** method provides an efficient means for retrieving all columns, rather than retrieving each column individually. An array of **System.Object** to copy the attribute columns into.

#### GetString

[C#] public string GetString(int i);

[C++] public: \_\_sealed String\* GetString(int i);

[VB] NotOverridable Public Function GetString(ByVal i As Integer) As String

[JScript] public function GetString(i : int) : String;

## Description

Gets the value of the specified column as a string.

**Return Value:** The value of the specified column.

No conversions are performed, therefore the data retrieved must already be a string. The zero-based column ordinal.

## GetValue

```
[C#]          public          object          GetValue(int          i);
```

```
[C++]      public:      __sealed      Object*      GetValue(int          i);
```

```
[VB] NotOverridable Public Function GetValue(ByVal i As Integer) As Object
```

```
[JScript]    public    function    GetValue(i    :    int)    :    Object;
```

## Description

Gets the value of the specified column in its native format.

**System.Data.SqlClient.SqlDataReader.GetValue(System.Int32)** returns data using the .NET Framework types. The zero-based column ordinal.

## GetValues

```
[C#]          public          int          GetValues(object[]          values);
```

```
[C++]      public:      __sealed      int      GetValues(Object*      values      __gc[]);
```

```
[VB] NotOverridable Public Function GetValues(ByVal values() As Object) As Integer
```

```
[JScript]    public    function    GetValues(values    :    Object[])    :    int;
```

## Description

Gets all attribute columns in the collection for the current row.

**Return Value:** The number of instances of **System.Object** in the array.

For most applications, this method provides an efficient means for retrieving all columns, rather than retrieving each column individually. An array of **System.Object** into which to copy the attribute columns.

## IsDBNull

[C#] public bool IsDBNull(int i);

[C++] public: \_\_sealed bool IsDBNull(int i);

[VB] NotOverridable Public Function IsDBNull(ByVal i As Integer) As Boolean

[JScript] public function IsDBNull(i : int) : Boolean;

## Description

Gets a value indicating whether the column contains non-existent or missing values.

**Return Value:** **true** if the specified column value is equivalent to **System.DBNull**; otherwise, **false**. The zero-based column ordinal.

## NextResult

[C#] public bool NextResult();

[C++] public: \_\_sealed bool NextResult();

[VB] NotOverridable Public Function NextResult() As Boolean

[JScript] public function NextResult() : Boolean;



```

1  IEnumerable.GetEnumerator
2  [JScript] function IEnumerable.GetEnumerator() : IEnumerator;
3      IDisposable.Dispose
4
5  [C#]                void                IDisposable.Dispose();
6  [C++]                void                IDisposable::Dispose();
7  [VB]      Sub      Dispose()      Implements      IDisposable.Dispose
8  [JScript] function IDisposable.Dispose();
9      SQLDebugging class (System.Data.SqlClient)
10     ToString
11
12
13  Description
14     Included to support debugging applications. Not intended for direct use.
15     SQLDebugging
16     Example Syntax:
17     ToString
18
19  [C#]                public                SQLDebugging();
20  [C++]                public:                SQLDebugging();
21  [VB]      Public      Sub      New()
22  [JScript] public function SQLDebugging();
23      ISQLDebug.SQLDebug
24
25  [C#] bool ISQLDebug.SQLDebug(int dwpidDebugger, int dwpidDebuggee, string

```

```

1 pszMachineName, string pszSDIDLLName, int dwOption, int cbData, byte[]
2 rgbData);
3 [C++] bool ISQLDebug::SQLDebug(int dwpidDebugger, int dwpidDebuggee,
4 String* pszMachineName, String* pszSDIDLLName, int dwOption, int cbData,
5 unsigned char rgbData __gc[]);
6 [VB] Function SQLDebug(ByVal dwpidDebugger As Integer, ByVal
7 dwpidDebuggee As Integer, ByVal pszMachineName As String, ByVal
8 pszSDIDLLName As String, ByVal dwOption As Integer, ByVal cbData As
9 Integer, ByVal rgbData() As Byte) As Boolean Implements
10 ISQLDebug.SQLDebug
11 [JScript] function ISQLDebug.SQLDebug(dwpidDebugger : int, dwpidDebuggee :
12 int, pszMachineName : String, pszSDIDLLName : String, dwOption : int, cbData :
13 int, rgbData : Byte[]) : Boolean;
14     SqlError class (System.Data.SqlClient)
15     ToString
16
17
18
19
20
21
22
23
24
25

```

### *Description*

Collects information relevant to a warning or error returned by SQL Server.

This class cannot be inherited.

This class is created by the SQL Server .NET Data Provider when an error occurs. An instance of **System.Data.SqlClient.SqlError** is created and managed by the **System.Data.SqlClient.SqlErrorCollection** , which in turn is created by the **System.Data.SqlClient.SqlException** class.

### *Class*

1	ToString						
2							
3	[C#]	public	byte	Class	{get;}		
4	[C++]	public:	__property	unsigned	char	get_Class();	
5	[VB]	Public	ReadOnly	Property	Class	As	Byte
6	[JScript]	public	function	get	Class()	:	Byte;

7

8 *Description*

9 Gets the severity level of the error returned from SQL Server.

10 Messages with a severity level of 10 or less are informational and indicate

11 problems caused by mistakes in information that a user has entered. Severity

12 levels from 11 through 16 are generated by the user, and can be corrected by the

13 user. Severity levels from 17 through 25 indicate software or hardware errors.

14 When a level 17, 18, or 19 error occurs, you can continue working, although you

15 might not be able to execute a particular statement.

16	LineNumber						
17	ToString						
18							
19	[C#]	public	int	LineNumber	{get;}		
20	[C++]	public:	__property	int	get_LineNumber();		
21	[VB]	Public	ReadOnly	Property	LineNumber	As	Integer
22	[JScript]	public	function	get	LineNumber()	:	int;

23

24 *Description*

25

0000560-0404

1       Bets the line number within the Transact-SQL command batch or stored  
2 procedure that contains the error.

3       Line numbering starts at 1. If the value is 0, the line number is not  
4 applicable.

5       Message  
6       ToString

7  
8 [C#]           public           string           Message           {get;}  
9 [C++]          public:          \_\_property       String\*        get\_Message();  
10 [VB]   Public    ReadOnly    Property    Message    As    String  
11 [JScript]   public    function   get    Message()   :    String;

12  
13 *Description*

14       Gets the text describing the error.

15       Number  
16       ToString

17  
18 [C#]           public           int            Number           {get;}  
19 [C++]          public:          \_\_property       int         get\_Number();  
20 [VB]   Public    ReadOnly    Property    Number    As    Integer  
21 [JScript]   public    function   get    Number()   :    int;

22  
23 *Description*

24       Gets a number that identifies the type of error.

25       This number corresponds to an entry in the **master.dbo.sysmessages** table.



1	[C++]	public:	__property	String*	get_Source();
2	[VB]	Public	ReadOnly	Property	Source As String
3	[JScript]	public	function	get	Source() : String;

### Description

Gets the name of the provider that generated the error.

## State

ToString

10	[C#]	public	byte	State	{get;}
11	[C++]	public:	__property	unsigned char	get_State();
12	[VB]	Public	ReadOnly	Property	State As Byte
13	[JScript]	public	function	get State()	: Byte;

### Description

Gets the number modifying the error to provide additional information.

ToString

19	[C#]	public	override	string	ToString();		
20	[C++]	public:		String*	ToString();		
21	[VB]	Overrides	Public	Function	ToString()	As	String
22	[JScript]	public	override	function	ToString()	:	String;

### Description



Gets the number of errors in the collection.

Item

ToString

```
[C#]      public      SqlError      this[int      index]      {get;}
```

```
[C++]     public:     __property     SqlError*     get_Item(int     index);
```

```
[VB] Public Default ReadOnly Property Item(ByVal index As Integer) As  
SqlError
```

```
[JScript]     returnValue     =     SqlErrorCollectionObject.Item(index);
```

#### *Description*

Gets the error at the specified index. The zero-based index of the error to retrieve.

CopyTo

```
[C#]      public      void      CopyTo(Array      array,      int      index);
```

```
[C++]     public:     __sealed     void     CopyTo(Array*     array,     int     index);
```

```
[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As  
Integer)
```

```
[JScript]     public     function     CopyTo(array : Array, index : int);
```

#### *Description*

Copies the elements of the **System.Data.SqlClient.SqlErrorCollection** collection into an **System.Array** , starting at the given index within the

1 **System.Array** . The **System.Array** to copy elements into. The index from which  
2 to start copying into the *array* parameter.

3 GetEnumerator

4  
5 [C#] public IEnumerator GetEnumerator();

6 [C++] public: \_\_sealed IEnumerator\* GetEnumerator();

7 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

8 [JScript] public function GetEnumerator() : IEnumerator;

9  
10 *Description*

11 used to support the VB For Each ... Next syntax. not explicitly called.

12 SqlException class (System.Data.SqlClient)

13 ToString

14  
15  
16 *Description*

17 The exception that is thrown when a warning or error is returned by SQL  
18 Server. This class cannot be inherited.

19 This class is created whenever the SQL Server .NET Data Provider  
20 encounters a situation that it cannot handle. It always contains at least one instance  
21 of **System.Data.SqlClient.SqlError** .

22 Class

23 ToString

24  
25 [C#] public byte Class {get;}

```

1 [C++]      public:      __property      unsigned      char      get_Class();
2 [VB]      Public      ReadOnly      Property      Class      As      Byte
3 [JScript]      public      function      get      Class()      :      Byte;

```

#### 5 *Description*

6 Gets the severity level of the error returned from the SQL Server .NET  
7 Data Provider.

8 Messages with a severity level of 10 or less are informational and indicate  
9 problems caused by mistakes in information that a user has entered. Severity  
10 levels from 11 through 16 are generated by the user, and can be corrected by the  
11 user. Severity levels from 17 through 25 indicate software or hardware errors.  
12 When a level 17, 18, or 19 error occurs, you can continue working, although you  
13 might not be able to execute a particular statement.

14 Errors

15 ToString

```

17 [C#]      public      SqlErrorCollection      Errors      {get;}
18 [C++]      public:      __property      SqlErrorCollection*      get_Errors();
19 [VB]      Public      ReadOnly      Property      Errors      As      SqlErrorCollection
20 [JScript]      public      function      get      Errors()      :      SqlErrorCollection;

```

#### 22 *Description*

23 Gets a collection of one or more **System.Data.SqlClient.SqlError** objects  
24 that give detailed information about exceptions generated by the SQL Server .NET  
25 Data Provider.

The **System.Data.SqlClient.SqlErrorCollection** class always contains at least one instance of the **System.Data.SqlClient.SqlError** class.

HelpLink

HResult

InnerException

LineNumber

ToString

#### *Description*

Gets the line number within the Transact-SQL command batch or stored procedure that generated the error.

The line numbering starts at 1; if 0 the line number is not applicable.

Message

ToString

[C#]        public        override        string        Message        {get;}

[C++]     public:     \_\_property     virtual     String\*     get\_Message();

[VB]   Overrides   Public   ReadOnly   Property   Message   As   String

[JScript]     public     function     get     Message()     :     String;

#### *Description*

Gets the text describing the error.

1 This is a wrapper for the **System.Data.SqlClient.SqlError.Message**  
 2 property of the first **System.Data.SqlClient.SqlError** in the  
 3 **System.Data.SqlClient.SqlException.Errors** property.

4 Number

5 ToString

6  
 7 [C#] public int Number {get;}

8 [C++] public: \_\_property int get\_Number();

9 [VB] Public ReadOnly Property Number As Integer

10 [JScript] public function get Number() : int;

11  
 12 *Description*

13 Gets a number that identifies the type of error.

14 This number corresponds to an entry in the **master.dbo.sysmessages** table.

15 Procedure

16 ToString

17  
 18 [C#] public string Procedure {get;}

19 [C++] public: \_\_property String\* get\_Procedure();

20 [VB] Public ReadOnly Property Procedure As String

21 [JScript] public function get Procedure() : String;

22  
 23 *Description*

24 Gets the name of the stored procedure or remote procedure call (RPC) that  
 25 generated the error.

1 This is a wrapper for the **System.Data.SqlClient.SqlError.Procedure**  
 2 property of the first **System.Data.SqlClient.SqlError** in the  
 3 **System.Data.SqlClient.SqlException.Errors** property.

4 Server  
 5 ToString

6  
 7 [C#] public string Server {get;}  
 8 [C++] public: \_\_property String\* get\_Server();  
 9 [VB] Public ReadOnly Property Server As String  
 10 [JScript] public function get Server() : String;

11  
 12 *Description*

13 Gets the name of the computer running an instance of SQL Server that  
 14 generated the error.

15 This is a wrapper for the **System.Data.SqlClient.SqlError.Server**  
 16 property of the first **System.Data.SqlClient.SqlError** in the  
 17 **System.Data.SqlClient.SqlException.Errors** property.

18 Source  
 19 ToString

20  
 21 [C#] public override string Source {get;}  
 22 [C++] public: \_\_property virtual String\* get\_Source();  
 23 [VB] Overrides Public ReadOnly Property Source As String  
 24 [JScript] public function get Source() : String;

25

1  
2 *Description*

3 Gets the name of the provider that generated the error.

4 This is a wrapper for the **System.Data.SqlClient.SqlError.Source**  
5 property of the first **System.Data.SqlClient.SqlError** in the  
6 **System.Data.SqlClient.SqlException.Errors** property.

7 StackTrace

8 State

9 ToString

10  
11  
12 *Description*

13 Gets the number modifying the error to provide additional information.

14 This is a wrapper for the **System.Data.SqlClient.SqlError.State** property  
15 of the first **System.Data.SqlClient.SqlError** in the  
16 **System.Data.SqlClient.SqlException.Errors** property.

17 TargetSite

18 **ISerializable.GetObjectData**

19  
20 [C#] void **ISerializable.GetObjectData**(**SerializationInfo** si, **StreamingContext**  
21 context);

22 [C++] void **ISerializable::GetObjectData**(**SerializationInfo\*** si, **StreamingContext**  
23 context);

24 [VB] Sub **GetObjectData**(ByVal si As **SerializationInfo**, ByVal context As  
25 **StreamingContext**) Implements **ISerializable.GetObjectData**

1 [JScript] function ISerializable.GetObjectData(si : SerializationInfo, context :  
2 StreamingContext);

3       SqlInfoMessageEventArgs class (System.Data.SqlClient)

4       ToString

5

6

7 *Description*

8       Provides data for the **System.Data.SqlClient.SqlConnection.InfoMessage**  
9 event. This class cannot be inherited.

10       The **System.Data.SqlClient.SqlConnection.InfoMessage** event contains a  
11 **System.Data.SqlClient.SqlErrorCollection** collection which contains the  
12 warnings sent from the server.

13       Errors

14       ToString

15

16 [C#]       public       SqlErrorCollection       Errors       {get;}

17 [C++]     public:     \_\_property     SqlErrorCollection\*     get\_Errors();

18 [VB]     Public     ReadOnly     Property     Errors     As     SqlErrorCollection

19 [JScript]     public     function     get     Errors()     :     SqlErrorCollection;

20

21 *Description*

22       Gets the collection of warnings sent from the server.

23       SqlInfoMessageEventHandler delegate (System.Data.SqlClient)

24       ToString

25

1  
2  
3 *Description*

4 Represents the method that will handle the  
5 **System.Data.SqlClient.SqlConnection.InfoMessage** event of a  
6 **System.Data.SqlClient.SqlConnection** . The source of the event. A  
7 **System.Data.SqlClient.SqlInfoMessageEventArgs** object that contains the event  
8 data.

9 When you create a **System.Data.SqlClient.SqlInfoMessageEventArgs**  
10 delegate, you identify the method that will handle the event. To associate the event  
11 with your event handler, add an instance of the delegate to the event. The event  
12 handler is called whenever the event occurs, unless you remove the delegate. For  
13 more information about event handler delegates, see .

14 **SqlParameter** class (System.Data.SqlClient)

15 **ToString**

16  
17  
18 *Description*

19 Represents a parameter to a **System.Data.SqlClient.SqlCommand** , and  
20 optionally, its mapping to **System.Data.DataSet** columns. This class cannot be  
21 inherited.

22 Parameter names are not case sensitive.

23 **SqlParameter**

24 *Example Syntax:*

25 **ToString**

```

1
2 [C#] public SqlParameter();
3 [C++] public: SqlParameter();
4 [VB] Public Sub New()
5 [JScript] public function SqlParameter(); Initializes a new instance of the
6 System.Data.SqlClient.SqlParameter class.

```

7

8 *Description*

9 Initializes a new instance of the **System.Data.SqlClient.SqlParameter**

10 class.

11 SqlParameter

12 *Example Syntax:*

13 ToString

```

14
15 [C#] public SqlParameter(string parameterName, object value);
16 [C++] public: SqlParameter(String* parameterName, Object* value);
17 [VB] Public Sub New(ByVal parameterName As String, ByVal value As Object)
18 [JScript] public function SqlParameter(parameterName : String, value : Object);

```

19

20 *Description*

21 Initializes a new instance of the **System.Data.SqlClient.SqlParameter**

22 class with the parameter name and a **System.Data.SqlClient.SqlParameter**

23 object. The name of the parameter to map. An **System.Object** that is the value of

24 the **System.Data.SqlClient.SqlParameter**.

25 SqlParameter

*Example Syntax:*

*ToString*

```
[C#] public SqlParameter(string parameterName, SqlDbType dbType);  
[C++] public: SqlParameter(String* parameterName, SqlDbType dbType);  
[VB] Public Sub New(ByVal parameterName As String, ByVal dbType As  
SqlDbType)  
[JScript] public function SqlParameter(parameterName : String, dbType :  
SqlDbType);
```

### *Description*

Initializes a new instance of the **System.Data.SqlClient.SqlParameter** class with the parameter name and the data type.

The data type, and if appropriate, **System.Data.OleDb.OleDbParameter.Size** and **System.Data.OleDb.OleDbParameter.Precision** are inferred from the value of the *dbType* parameter. The name of the parameter to map. One of the **System.Data.SqlDbType** values.

**SqlParameter**

*Example Syntax:*

*ToString*

```
[C#] public SqlParameter(string parameterName, SqlDbType dbType, int size);  
[C++] public: SqlParameter(String* parameterName, SqlDbType dbType, int  
size);
```

```

1 [VB] Public Sub New(ByVal parameterName As String, ByVal dbType As
2 SqlDbType,           ByVal           size           As           Integer)
3 [JScript] public function SqlParameter(parameterName : String, dbType :
4 SqlDbType,           size           :           int);

```

#### *Description*

Initializes a new instance of the **System.Data.SqlClient.SqlParameter** class with the parameter name, the **System.Data.SqlDbType** , and the size.

The **System.Data.OleDb.OleDbParameter.Size** is inferred from the value of the *dbType* parameter if it is not explicitly set in the *size* parameter. The name of the parameter to map. One of the **System.Data.SqlDbType** values. The width of the parameter.

**SqlParameter**

*Example Syntax:*

**ToString**

```

17 [C#] public SqlParameter(string parameterName, SqlDbType dbType, int size,
18 string                                sourceColumn);
19 [C++] public: SqlParameter(String* parameterName, SqlDbType dbType, int size,
20 String*                                sourceColumn);
21 [VB] Public Sub New(ByVal parameterName As String, ByVal dbType As
22 SqlDbType, ByVal size As Integer, ByVal sourceColumn As String)
23 [JScript] public function SqlParameter(parameterName : String, dbType :
24 SqlDbType, size : int, sourceColumn : String);

```

1  
2 *Description*

3        Initializes a new instance of the **System.Data.SqlClient.SqlParameter**  
4 class with the parameter name, the **System.Data.SqlDbType** , the size, the source  
5 column name, and a **System.Data.DataRowVersion** to use.

6        The **System.Data.OleDb.OleDbParameter.Size** is inferred from the value  
7 of the *dbType* parameter if it is not explicitly set in the *size* parameter. The name of  
8 the parameter to map. One of the **System.Data.SqlDbType** values. The width of  
9 the parameter. The name of the source column.

10        SqlParameter

11        *Example Syntax:*

12        ToString

13  
14 [C#] public SqlParameter(string parameterName, SqlDbType dbType, int size,  
15 ParameterDirection direction, bool isNullable, byte precision, byte scale, string  
16 sourceColumn, DataRowVersion sourceVersion, object value);  
17 [C++] public: SqlParameter(String\* parameterName, SqlDbType dbType, int size,  
18 ParameterDirection direction, bool isNullable, unsigned char precision, unsigned  
19 char scale, String\* sourceColumn, DataRowVersion sourceVersion, Object\*  
20 value);

21 [VB] Public Sub New(ByVal parameterName As String, ByVal dbType As  
22 SqlDbType, ByVal size As Integer, ByVal direction As ParameterDirection,  
23 ByVal isNullable As Boolean, ByVal precision As Byte, ByVal scale As Byte,  
24 ByVal sourceColumn As String, ByVal sourceVersion As DataRowVersion,  
25 ByVal value As Object)

```

1 [JScript] public function SqlParameter(parameterName : String, dbType :
2 SqlDbType, size : int, direction : ParameterDirection, isNullable : Boolean,
3 precision : Byte, scale : Byte, sourceColumn : String, sourceVersion :
4 DataRowVersion, value : Object);
5

```

#### 6 *Description*

7        Initializes a new instance of the **System.Data.SqlClient.SqlParameter**  
8 class with the parameter name, the type of the parameter, the size of the parameter,  
9 a **System.Data.ParameterDirection** , the precision of the parameter, the scale of  
10 the parameter, the source column, a **System.Data.DataRowVersion** to use, and  
11 the value of the parameter.

12        The **System.Data.OleDb.OleDbParameter.Size** and  
13 **System.Data.OleDb.OleDbParameter.Precision** are inferred from the value of  
14 the *dbType* parameter if they are not explicitly set in the *size* and *precision*  
15 parameters. The name of the parameter to map. One of the  
16 **System.Data.SqlDbType** values. The width of the parameter. One of the  
17 **System.Data.ParameterDirection** values. **true** if the value of the field can be  
18 null, otherwise **false**. The total number of digits to the left and right of the decimal  
19 point to which **System.Data.SqlClient.SqlParameter.Value** is resolved. The total  
20 number of decimal places to which **System.Data.SqlClient.SqlParameter.Value**  
21 is resolved. The name of the source column. One of the  
22 **System.Data.DataRowVersion** values. An **System.Object** that is the value of the  
23 **System.Data.SqlClient.SqlParameter**.

24        DbType

25        ToString

```

1
2 [C#]      public      DbType      DbType      {get;      set;}
3 [C++] public: __property DbType get_DbType();public: __property void
4 set_DbType(DbType);
5 [VB]      Public      Property      DbType      As      DbType
6 [JScript] public function get DbType() : DbType;public function set
7 DbType(DbType);
8

```

### 9 *Description*

10 Gets or sets the **System.Data.DbType** of the parameter.

11 The **System.Data.SqlClient.SqlParameter.SqlDbType** and  
12 **System.Data.SqlClient.SqlParameter.DbType** are linked. Therefore, setting the  
13 **System.Data.SqlClient.SqlParameter.DbType** changes the  
14 **System.Data.SqlClient.SqlParameter.SqlDbType** to a supporting  
15 **System.Data.SqlClient.SqlParameter.SqlDbType**.

16 Direction

17 ToString

```

18
19 [C#]      public      ParameterDirection      Direction      {get;      set;}
20 [C++] public: __property ParameterDirection get_Direction();public: __property
21 void      set_Direction(ParameterDirection);
22 [VB]      Public      Property      Direction      As      ParameterDirection
23 [JScript] public function get Direction() : ParameterDirection;public function set
24 Direction(ParameterDirection);
25

```

If the **System.Data.ParameterDirection** is output, and execution of the associated **System.Data.SqlClient.SqlCommand** does not return a value, the **System.Data.SqlClient.SqlParameter** contains a null value.

## ToString

```
[C++] public: __property bool get_IsNullable();public: __property void
set_IsNullable(bool);
```

```
[JScript] public function get IsNullable() : Boolean;public function set
IsNullable(Boolean);
```

## ToString

```
[C++] public: __property int get Offset();public: __property void set_Offset(int);
```

```

1  [VB]          Public          Property          Offset          As          Integer
2  [JScript] public function get Offset() : int;public function set Offset(int);

```

3

4 *Description*

5 Gets or sets the offset to the **System.Data.SqlClient.SqlParameter.Value**

6 property.

7 This property is used for binary and string types. It returns the number of

8 bytes for binary types, and the number of characters for strings. The count for

9 strings does not include the terminating character, if **null** .

10 ParameterName

11 ToString

12

```

13 [C#]          public          string          ParameterName          {get;          set;}
14 [C++] public: __property String* get_ParameterName();public: __property void
15 set_ParameterName(String*);

```

```

16 [VB]          Public          Property          ParameterName          As          String
17 [JScript] public function get ParameterName() : String;public function set
18 ParameterName(String);

```

19

20 *Description*

21 Gets or sets the name of the **System.Data.SqlClient.SqlParameter** .

22 The **System.Data.SqlClient.SqlParameter.ParameterName** is specified

23 in the form **@paramname**. You must set

24 **System.Data.SqlClient.SqlParameter.ParameterName** before executing a

25 **System.Data.SqlClient.SqlCommand** that relies on parameters.

```

1      Precision
2      ToString
3
4  [C#]      public      byte      Precision      {get;      set;}
5  [C++] public: __property unsigned char get_Precision();public: __property void
6  set_Precision(unsigned char);
7  [VB]      Public      Property      Precision      As      Byte
8  [JScript] public function get Precision() : Byte;public function set Precision(Byte);
9

```

#### 10 *Description*

11 Gets or sets the maximum number of digits used to represent the  
12 **System.Data.SqlClient.SqlParameter.Value** property.

13 The **System.Data.SqlClient.SqlParameter.Precision** property is used by  
14 parameters which have a **System.Data.SqlDbType** of **Decimal** .

15 Scale

16 ToString

```

17
18 [C#]      public      byte      Scale      {get;      set;}
19 [C++] public: __property unsigned char get_Scale();public: __property void
20 set_Scale(unsigned char);
21 [VB]      Public      Property      Scale      As      Byte
22 [JScript] public function get Scale() : Byte;public function set Scale(Byte);
23

```

#### 24 *Description*

25

1 Gets or sets the number of decimal places to which  
2 **System.Data.SqlClient.SqlParameter.Value** is resolved.

3 The **System.Data.SqlClient.SqlParameter.Scale** property is used by  
4 parameters which have a **System.Data.SqlDbType** of **Decimal** .

5 Size

6 ToString

7  
8 [C#] public int Size {get; set;}

9 [C++] public: \_\_property int get\_Size();public: \_\_property void set\_Size(int);

10 [VB] Public Property Size As Integer

11 [JScript] public function get Size() : int;public function set Size(int);

12  
13 *Description*

14 Gets or sets the maximum size, in bytes, of the data within the column.

15 The **System.Data.SqlClient.SqlParameter.Size** property is used for binary  
16 and string types.

17 SourceColumn

18 ToString

19  
20 [C#] public string SourceColumn {get; set;}

21 [C++] public: \_\_property String\* get\_SourceColumn();public: \_\_property void  
22 set\_SourceColumn(String\*);

23 [VB] Public Property SourceColumn As String

24 [JScript] public function get SourceColumn() : String;public function set  
25 SourceColumn(String);

1  
2 *Description*

3 Gets or sets the name of the source column that is mapped to the  
4 **System.Data.DataSet** and used for loading or returning the  
5 **System.Data.SqlClient.SqlParameter.Value** .

6 The link between the value of the **System.Data.SqlClient.SqlParameter**  
7 and the **System.Data.DataTable** may be bidirectional depending on the value of  
8 the **System.Data.SqlClient.SqlParameter.Direction** property.

9 SourceVersion

10 ToString

11  
12 [C#] public DataRowVersion SourceVersion {get; set;}  
13 [C++] public: \_\_property DataRowVersion get\_SourceVersion();public:  
14 \_\_property void set\_SourceVersion(DataRowVersion);  
15 [VB] Public Property SourceVersion As DataRowVersion  
16 [JScript] public function get SourceVersion() : DataRowVersion;public function  
17 set SourceVersion(DataRowVersion);  
18

19 *Description*

20 Gets or sets the **System.Data.DataRowVersion** to use when loading  
21 **System.Data.SqlClient.SqlParameter.Value** .

22 This property is used by the  
23 **System.Data.SqlClient.SqlDataAdapter.UpdateCommand** during the  
24 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to  
25 determine whether the original or current value is used for a parameter value. This

allows primary keys to be updated. This property is ignored by the **System.Data.SqlClient.SqlDataAdapter.InsertCommand** and **System.Data.SqlClient.SqlDataAdapter.DeleteCommand**. This property is set to the version of the **System.Data.DataRow** used by the **System.Data.DataRow.Item(System.Int32)** property, or the **System.Data.DataRow.GetChildRows(System.String)** method of the **System.Data.DataRow** object.

**SqlDbType**  
**ToString**

[C#]        public        SqlDbType        SqlDbType        {get;        set;}  
[C++] public: \_\_property SqlDbType get\_SqlDbType();public: \_\_property void set\_SqlDbType(SqlDbType);  
[VB]        Public        Property        SqlDbType        As        SqlDbType  
[JScript] public function get SqlDbType() : SqlDbType;public function set SqlDbType(SqlDbType);

#### *Description*

Gets or sets the **System.Data.SqlDbType** of the parameter.  
The **System.Data.SqlClient.SqlParameter.SqlDbType** and **System.Data.SqlClient.SqlParameter.DbType** are linked. Therefore, setting the **System.Data.SqlClient.SqlParameter.DbType** changes the **System.Data.SqlClient.SqlParameter.SqlDbType** to a supporting **System.Data.SqlDbType**.

**Value**

```

1      ToString
2
3  [C#]      public      object      Value      {get;      set;}
4  [C++]  public:  __property  Object*  get_Value();public:  __property  void
5  set_Value(Object*);
6  [VB]      Public      Property      Value      As      Object
7  [JScript] public function get Value() : Object;public function set Value(Object);
8

```

### 9 *Description*

10 Gets or sets the value of the parameter.

11 For input parameters, the value is bound to the  
12 **System.Data.SqlClient.SqlCommand** that is sent to the server. For output and  
13 return value parameters, the value is set on completion of the  
14 **System.Data.SqlClient.SqlCommand** and after the  
15 **System.Data.SqlClient.SqlDataReader** is closed.

### 16 **ICloneable.Clone**

```

17
18 [C#]      object      ICloneable.Clone();
19 [C++]      Object*      ICloneable::Clone();
20 [VB]  Function  Clone()  As  Object  Implements  ICloneable.Clone
21 [JScript] function ICloneable.Clone() : Object;

```

### 22 **ToString**

```

23
24 [C#]      public      override      string      ToString();
25 [C++]      public:      String*      ToString();

```

1 [VB] Overrides Public Function ToString() As String  
 2 [JScript] public override function ToString() : String;

4 *Description*

5 Gets a string containing the  
 6 **System.Data.SqlClient.SqlParameter.ParameterName**.

7 *Return Value:* A string containing the  
 8 **System.Data.SqlClient.SqlParameter.ParameterName**.

9 SqlParameterCollection class (System.Data.SqlClient)

10 ToString

13 *Description*

14 Collects all parameters relevant to a **System.Data.SqlClient.SqlCommand**  
 15 and their respective mappings to **System.Data.DataSet** columns. This class  
 16 cannot be inherited.

17 The number of the parameters in the collection must be equal to the number  
 18 of parameter placeholders within the command text, or SQL Server raises an error.

19 Count

20 ToString

22 [C#] public int Count {get;}

23 [C++] public: \_\_property int get\_Count();

24 [VB] Public ReadOnly Property Count As Integer

25 [JScript] public function get Count() : int;

## *Description*

Gets the number of **System.Data.SqlClient.SqlParameter** objects in the collection.

Item

ToString

```
[C#]      public      SqlParameter      this[int      index]      {get;      set;}
[C++] public: __property SqlParameter* get_Item(int index);public: __property
void      set_Item(int      index,      SqlParameter*);
[VB] Public Default Property Item(ByVal index As Integer) As SqlParameter
[JScript]      returnValue      =
SqlParameterCollectionObject.Item(index);SqlParameterCollectionObject.Item(in
dex) = returnValue; Gets the System.Data.SqlClient.SqlParameter with a
specified      attribute.
```

## *Description*

Gets the **System.Data.SqlClient.SqlParameter** at the specified index. The zero-based index of the parameter to retrieve.

Item

ToString

```
[C#]      public      SqlParameter      this[string      parameterName]      {get;      set;}
[C++] public:      __property      SqlParameter*      get_Item(String*
parameterName);public:      __property      void      set_Item(String*      parameterName,
```

```

1  SqlParameter*);
2  [VB] Public Default Property Item(ByVal parameterName As String) As
3  SqlParameter
4  [JScript]                                     returnValue =
5  SqlParameterCollectionObject.Item(parameterName);SqlParameterCollectionObje
6  ct.Item(parameterName) = returnValue;
7

```

### *Description*

Gets the **System.Data.SqlClient.SqlParameter** with the specified name.  
The name of the parameter to retrieve.

### *Add*

```

13 [C#]          public          int          Add(object          value);
14 [C++]         public:         __sealed     int          Add(Object*      value);
15 [VB] NotOverridable Public Function Add(ByVal value As Object) As Integer
16 [JScript]     public  function  Add(value : Object) : int; Adds a
17 System.Data.SqlClient.SqlParameter          to          the
18 System.Data.SqlClient.SqlParameterCollection
19

```

### *Description*

Adds the specified **System.Data.SqlClient.SqlParameter** object to the  
**System.Data.SqlClient.SqlParameterCollection**.  
*Return Value:* A reference to the new **System.Data.SqlClient.SqlParameter**  
object. The **System.Data.SqlClient.SqlParameter** to add to the collection.

### *Add*

```

1
2 [C#]      public      SqlParameter      Add(SqlParameter      value);
3 [C++]     public:     SqlParameter*     Add(SqlParameter*     value);
4 [VB] Public Function Add(ByVal value As SqlParameter) As SqlParameter
5 [JScript] public function Add(value : SqlParameter) : SqlParameter;
6

```

### *Description*

Adds the specified **System.Data.SqlClient.SqlParameter** object to the **System.Data.SqlClient.SqlCommand**.

*Return Value:* A reference to the new **System.Data.SqlClient.SqlParameter** object. The **System.Data.SqlClient.SqlParameter** to be added.

### *Add*

```

14 [C#] public SqlParameter Add(string parameterName, object value);
15 [C++] public: SqlParameter* Add(String* parameterName, Object* value);
16 [VB] Public Function Add(ByVal parameterName As String, ByVal value As
17 Object) As SqlParameter
18 [JScript] public function Add(parameterName : String, value : Object) :
19 SqlParameter;
20

```

### *Description*

Adds a **System.Data.SqlClient.SqlParameter** to the **System.Data.SqlClient.SqlParameterCollection** with the specified parameter name and **System.Data.SqlClient.SqlParameter** object.

1 *Return Value:* A reference to the new **System.Data.SqlClient.SqlParameter**  
2 object.

3 When you specify **System.DBNull.Value** in the *value* parameter, you  
4 should also explicitly set the **System.Data.SqlClient.SqlParameter.SqlDbType**  
5 as demonstrated in this C# example: `SqlCommand rComm = new`  
6 `SqlCommand(null, rConn); rComm.CommandText = "insert into mytable values`  
7 `(?); rComm.Parameters.Add("@p1", DBNull.Value);`  
8 `rComm.Parameters["@p1"].SqlDbType = SqlDbType.Integer;`x The  
9 **System.Data.SqlClient.SqlParameter.Value** of the  
10 **System.Data.SqlClient.SqlParameter** to add to the collection.

11 Add

12  
13 [C#] public SqlParameter Add(string parameterName, SqlDbType sqlDbType);

14 [C++] public: SqlParameter\* Add(String\* parameterName, SqlDbType  
15 sqlDbType);

16 [VB] Public Function Add(ByVal parameterName As String, ByVal sqlDbType  
17 As SqlDbType) As SqlParameter

18 [JScript] public function Add(parameterName : String, sqlDbType : SqlDbType) :  
19 SqlParameter;

20  
21 *Description*

22 Adds a **System.Data.SqlClient.SqlParameter** to the  
23 **System.Data.SqlClient.SqlParameterCollection** with the parameter name and  
24 the data type.



```

1  [VB] Public Function Add(ByVal parameterName As String, ByVal sqlDbType
2  As SqlDbType, ByVal size As Integer, ByVal sourceColumn As String) As
3  SqlParameter

```

```

4  [JScript] public function Add(parameterName : String, sqlDbType : SqlDbType,
5  size      :      int,      sourceColumn      :      String)      :      SqlParameter;

```

#### 7 *Description*

8 Adds a **System.Data.SqlClient.SqlParameter** to the

9 **System.Data.SqlClient.SqlParameterCollection** with the parameter name, the

10 data type, the column width, and the source column name.

11 *Return Value:* A reference to the new **System.Data.SqlClient.SqlParameter**

12 object. The width of the column. The name of the source column.

#### 13 **Clear**

```

15 [C#]          public          void          Clear();

```

```

16 [C++]          public:          __sealed          void          Clear();

```

```

17 [VB]          NotOverridable          Public          Sub          Clear()

```

```

18 [JScript]          public          function          Clear();

```

#### 20 *Description*

21 Removes all items from the collection.

#### 22 **Contains**

```

24 [C#]          public          bool          Contains(object          value);

```

```

25 [C++]          public:          __sealed          bool          Contains(Object*          value);

```

1 [VB] NotOverridable Public Function Contains(ByVal value As Object) As  
2 Boolean

3 [JScript] public function Contains(value : Object) : Boolean;

4  
5 *Description*

6 Indicates whether a **System.Data.SqlClient.SqlParameter** exists in the  
7 collection.

8 *Return Value:* **true** if the collection contains the  
9 **System.Data.SqlClient.SqlParameter** object; otherwise, **false** . A  
10 **System.Data.SqlClient.SqlParameter** object.

11 Contains

12  
13 [C#] public bool Contains(string value);

14 [C++] public: \_\_sealed bool Contains(String\* value);

15 [VB] NotOverridable Public Function Contains(ByVal value As String) As  
16 Boolean

17 [JScript] public function Contains(value : String) : Boolean; Indicates whether a  
18 **System.Data.SqlClient.SqlParameter** exists in the collection.

19  
20 *Description*

21 Indicates whether a **System.Data.SqlClient.SqlParameter** with the  
22 specified parameter name exists in the collection.

23 *Return Value:* **true** if the collection contains the parameter; otherwise, **false** . The  
24 name of the parameter to retrieve.

25 CopyTo

```

[C#]      public      void      CopyTo(Array      array,      int      index);
[C++]    public:      __sealed      void      CopyTo(Array*      array,      int      index);
[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As
Integer)
[JavaScript] public function CopyTo(array : Array, index : int);
    
```

### *Description*

Copies **System.Data.SqlClient.SqlParameter** objects from the **System.Data.SqlClient.SqlParameterCollection** to the specified array. An **System.Array** to which to copy the **System.Data.SqlClient.SqlParameter** objects in the collection. The starting index of the array.

### GetEnumerator

```

[C#]      public      IEnumerator      GetEnumerator();
[C++]    public:      __sealed      IEnumerator*      GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JavaScript] public function GetEnumerator() : IEnumerator;
    
```

### *Description*

### IndexOf

```

[C#]      public      int      IndexOf(object      value);
[C++]    public:      __sealed      int      IndexOf(Object*      value);
[VB] NotOverridable Public Function IndexOf(ByVal value As Object) As Integer
    
```

1 [JScript] public function IndexOf(value : Object) : int;

2

3 *Description*

4 Gets the location of a **System.Data.SqlClient.SqlParameter** in the  
5 collection.

6 *Return Value:* The location of the **System.Data.SqlClient.SqlParameter** in the  
7 collection.

8 IndexOf

9

10 [C#] public int IndexOf(string parameterName);

11 [C++] public: \_\_sealed int IndexOf(String\* parameterName);

12 [VB] NotOverridable Public Function IndexOf(ByVal parameterName As String)

13 As Integer

14 [JScript] public function IndexOf(parameterName : String) : int; Gets the location  
15 of a **System.Data.SqlClient.SqlParameter** in the collection.

16

17 *Description*

18 Gets the location of the **System.Data.SqlClient.SqlParameter** in the  
19 collection with a specific parameter name.

20 *Return Value:* The location of the **System.Data.SqlClient.SqlParameter** in the  
21 collection. The name of the parameter to retrieve.

22 Insert

23

24 [C#] public void Insert(int index, object value);

25 [C++] public: \_\_sealed void Insert(int index, Object\* value);

[VB] NotOverridable Public Sub Insert(ByVal index As Integer, ByVal value As Object)

[JScript] public function Insert(index : int, value : Object);

#### *Description*

Inserts a **System.Data.SqlClient.SqlParameter** in the collection at the specified index. The zero-based index within the collection to insert the *valueparameter*. The **System.Data.SqlClient.SqlParameter** to add to the collection.

#### **Remove**

[C#] public void Remove(object value);

[C++] public: \_\_sealed void Remove(Object\* value);

[VB] NotOverridable Public Sub Remove(ByVal value As Object)

[JScript] public function Remove(value : Object);

#### *Description*

Removes the specified **System.Data.SqlClient.SqlParameter** from the collection. A **System.Data.SqlClient.SqlParameter** object to remove from the collection.

#### **RemoveAt**

[C#] public void RemoveAt(int index);

[C++] public: \_\_sealed void RemoveAt(int index);

[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)

1 [JScript] public function RemoveAt(index : int); Removes the specified  
2 **System.Data.SqlClient.SqlParameter** from the collection.

3  
4 *Description*

5 Removes the specified **System.Data.SqlClient.SqlParameter** from the  
6 collection using a specific index. The zero-based index of the parameter.

7 RemoveAt

8  
9 [C#] public void RemoveAt(string parameterName);

10 [C++] public: \_\_sealed void RemoveAt(String\* parameterName);

11 [VB] NotOverridable Public Sub RemoveAt(ByVal parameterName As String)

12 [JScript] public function RemoveAt(parameterName : String);

13  
14 *Description*

15 Removes the specified **System.Data.SqlClient.SqlParameter** from the  
16 collection using the parameter name.

17 SqlParameterEventArgs class (System.Data.SqlClient)

18 ToString

19  
20  
21 *Description*

22 Provides data for the  
23 **System.Data.SqlClient.SqlDataAdapter.RowUpdated** event. This class cannot  
24 be inherited.

25

The **System.Data.SqlClient.SqlDataAdapter.RowUpdated** event is raised when an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to a row is completed.

**SqlRowUpdatedEventArgs**

*Example Syntax:*

**ToString**

[C#] public **SqlRowUpdatedEventArgs**(DataRow row, IDbCommand command, StatementType statementType, DataTableMapping tableMapping);

[C++] public: **SqlRowUpdatedEventArgs**(DataRow\* row, IDbCommand\* command, StatementType statementType, DataTableMapping\* tableMapping);

[VB] Public Sub New(ByVal row As DataRow, ByVal command As IDbCommand, ByVal statementType As StatementType, ByVal tableMapping As DataTableMapping)

[JScript] public function **SqlRowUpdatedEventArgs**(row : DataRow, command : IDbCommand, statementType : StatementType, tableMapping : DataTableMapping);

### *Description*

Initializes a new instance of the **System.Data.SqlClient.SqlRowUpdatedEventArgs** class. The **System.Data.DataRow** sent through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**. The **System.Data.IDbCommand** executed when

**System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** is called. One of the **System.Data.StatementType** values that specifies the type of query executed. The **System.Data.Common.DataTableMapping** sent through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**.

Command

ToString

[C#]        public        new        SqlCommand        Command        {get;}

[C++]       public:        \_\_property        SqlCommand\*        get\_Command();

[VB]        Public        ReadOnly        Property        Command        As        SqlCommand

[JScript]    public        function        get        Command()        :        SqlCommand;

### *Description*

Gets or sets the **System.Data.SqlClient.SqlCommand** executed when **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** is called.

Errors

RecordsAffected

Row

StatementType

Status

TableMapping

SqlRowUpdatedEventHandler delegate (System.Data.SqlClient)

ToString

1  
2  
3 *Description*

4 Represents the method that will handle the  
5 **System.Data.SqlClient.SqlDataAdapter.RowUpdated** event of a  
6 **System.Data.SqlClient.SqlDataAdapter** . The source of the event. The  
7 **System.Data.SqlClient.SqlRowUpdatedEventArgs** that contains the event data.

8 The handler is not required perform any action, and your code should avoid  
9 generating exceptions or allowing exceptions to propagate to the calling method.  
10 Any exceptions that do reach the caller are ignored.

11 **SqlRowUpdatingEventArgs** class (System.Data.SqlClient)

12 ToString

13  
14  
15 *Description*

16 Provides data for the  
17 **System.Data.SqlClient.SqlDataAdapter.RowUpdating** event. This class cannot  
18 be inherited.

19 The **System.Data.SqlClient.SqlDataAdapter.RowUpdating** event is  
20 raised before an  
21 **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)** to a  
22 row.

23 **SqlRowUpdatingEventArgs**

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public SqlRowUpdatingEventArgs(DataRow row, IDbCommand command,
3 StatementType statementType, DataTableMapping tableMapping);
4 [C++] public: SqlRowUpdatingEventArgs(DataRow* row, IDbCommand*
5 command, StatementType statementType, DataTableMapping* tableMapping);
6 [VB] Public Sub New(ByVal row As DataRow, ByVal command As
7 IDbCommand, ByVal statementType As StatementType, ByVal tableMapping As
8 DataTableMapping)
9 [JScript] public function SqlRowUpdatingEventArgs(row : DataRow, command :
10 IDbCommand, statementType : StatementType, tableMapping :
11 DataTableMapping);
12

```

### *Description*

Initializes a new instance of the **System.Data.SqlClient.SqlRowUpdatingEventArgs** class. The **System.Data.DataRow** to **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**. The **System.Data.IDbCommand** to execute during **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**. One of the **System.Data.StatementType** values that specifies the type of query executed. The **System.Data.Common.DataTableMapping** sent through an **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**.

Command  
 ToString

25

```

1
2 [C#]      public      new      SqlCommand      Command      {get;      set;}
3 [C++] public: __property SqlCommand* get_Command();public: __property void
4 set_Command(SqlCommand*);
5 [VB]      Public      Property      Command      As      SqlCommand
6 [JScript] public function get Command() : SqlCommand;public function set
7 Command(SqlCommand);
8

```

### *Description*

Gets or sets the **System.Data.SqlClient.SqlCommand** to execute when performing the **System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)**.

Errors

Row

StatementType

Status

TableMapping

SqlRowUpdatingEventHandler delegate (System.Data.SqlClient)

ToString

### *Description*

Represents the method that will handle the **System.Data.SqlClient.SqlDataAdapter.RowUpdating** event of a

1 **System.Data.SqlClient.SqlDataAdapter** . The source of the event. The  
2 **System.Data.SqlClient.SqlRowUpdatingEventArgs** that contains the event data.

3 The handler is not required perform any action, and your code should avoid  
4 generating exceptions or allowing exceptions to propagate to the calling method.  
5 Any exceptions that do reach the caller are ignored.

6 **SqlTransaction** class (**System.Data.SqlClient**)

7 **ToString**

8  
9  
10 *Description*

11 Represents a Transact-SQL transaction to be made in a SQL Server  
12 database. This class cannot be inherited.

13 The application creates a **System.Data.SqlClient.SqlTransaction** object  
14 by calling **System.Data.SqlClient.SqlConnection.BeginTransaction** on the  
15 **System.Data.SqlClient.SqlConnection** object. All subsequent operations  
16 associated with the transaction (for example, committing or aborting the  
17 transaction), are performed on the **System.Data.SqlClient.SqlTransaction** object.

18 **Connection**

19 **ToString**

20  
21 [C#] public SqlConnection Connection {get;}

22 [C++] public: \_\_property SqlConnection\* get\_Connection();

23 [VB] Public ReadOnly Property Connection As SqlConnection

24 [JScript] public function get Connection() : SqlConnection;

25 **IsolationLevel**

## ToString

```
[C#]      public      IsolationLevel      IsolationLevel      {get;}
[C++]     public:     __property      IsolationLevel      get_IsolationLevel();
[VB]     Public  ReadOnly  Property  IsolationLevel  As  IsolationLevel
[JScript] public  function  get  IsolationLevel()  :  IsolationLevel;
```

### *Description*

Specifies the **System.Data.IsolationLevel** for this transaction.

Parallel transactions are not supported. Therefore, the **System.Data.IsolationLevel** applies to the entire transaction.

## Commit

```
[C#]      public      void      Commit();
[C++]     public:     __sealed      void      Commit();
[VB]     NotOverridable      Public      Sub      Commit()
[JScript] public      function      Commit();
```

### *Description*

Commits the database transaction.

The **System.Data.SqlClient.SqlTransaction.Commit** method is equivalent to the Transact-Sql COMMIT TRANSACTION statement. For more information, see SQL Server Books Online.

## Dispose



## Rollback

```
[C#]      public      void      Rollback(string      transactionName);
[C++]     public:     void      Rollback(String*      transactionName);
[VB]     Public      Sub      Rollback(ByVal      transactionName      As      String)
[JScript] public function Rollback(transactionName : String); Rolls back a
transaction      from      a      pending      state.
```

### *Description*

Rolls back a transaction from a pending state, and specifies the transaction or savepoint name.

The **System.Data.SqlClient.SqlTransaction.Rollback** method is equivalent to the Transact-Sql ROLLBACK TRANSACTION statement. For more information, see SQL Server Books Online. The name of the transaction to rollback, or the savepoint to which to rollback.

## Save

```
[C#]      public      void      Save(string      savePointName);
[C++]     public:     void      Save(String*      savePointName);
[VB]     Public      Sub      Save(ByVal      savePointName      As      String)
[JScript] public function Save(savePointName      :      String);
```

### *Description*

Creates a savepoint in the transaction that can be used to roll back a portion of the transaction, and specifies the savepoint name.

**System.Data.SqlClient.SqlTransaction.Save(System.String)** method is equivalent to the Transact-SQL SAVE TRANSACTION statement. For more information, see SQL Server Books Online. The name of

### **System.Data.SqlTypes**

The namespace provides classes for native data types within SQL Server. These classes provide a safer, faster alternative to other data types. Using the objects within this namespace helps prevent type conversion errors caused in situations where loss of precision could occur. Because other data types are converted to and from SqlTypes behind the scenes, explicitly creating and using objects within this namespace results in faster code as well.

#### *Description*

The **System.Data.SqlTypes** namespace provides classes for native data types within SQL Server. These classes provide a safer, faster alternative to other data types. Using the objects within this namespace helps prevent type conversion errors caused in situations where loss of precision could occur. Because other data types are converted to and from SqlTypes behind the scenes, explicitly creating and using objects within this namespace results in faster code as well.

INullable interface (System.Data.SqlTypes)

#### *Description*

1 All of the **System.Data.SqlTypes** objects and structures implement the  
 2 **INullable** interface, reflecting the fact that, unlike the corresponding system types,  
 3 **SqlTypes** can legally contain the value null.

4 Properties:

5 **IsNull**

6						
7	[C#]	bool		IsNull		{get;}
8	[C++]		bool			get_IsNull();
9	[VB]	ReadOnly	Property	IsNull	As	Boolean
10	[JScript]	abstract	function	get	IsNull()	: Boolean;

11  
 12 *Description*

13 Indicates whether a structure is null.

14 **SqlBinary** structure (System.Data.SqlTypes)

15  
 16  
 17 *Description*

18 Represents a variable-length stream of binary data to be stored in or  
 19 retrieved from a database.

20						
21	[C#]	public	static	readonly	SqlBinary	Null;
22	[C++]	public:		static	SqlBinary	Null;
23	[VB]	Public	Shared	ReadOnly	Null	As SqlBinary
24	[JScript]	public	static	var	Null	: SqlBinary;

25

**U.S. DEPARTMENT OF AGRICULTURE**

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

6

7

8

9

11

12

2

14

15

17

18  
19  
20

21

23

24

25

```
[JScript]      public      function      get      IsNull()      :      Boolean;
```

### *Description*

Gets a value indicating whether whether the **System.Data.SqlTypes.SqlBinary.Value** property of the **System.Data.SqlTypes.SqlBinary** structure is null. This property is read-only.

Item

```
[C#]      public      byte      this[int      index]      {get;}
```

```
[C++]      public:      __property      unsigned      char      get_Item(int      index);
```

```
[VB] Public Default ReadOnly Property Item(ByVal index As Integer) As Byte
```

```
[JScript]      returnValue      =      SqlBinaryObject.Item(index);
```

### *Description*

Gets the single byte from the **Value** property located at the position indicated by the integer parameter, *index* . If *index* indicates a position beyond the end of the byte array, a **System.Data.SqlTypes.SqlNullValueException** will be raised. This property is read-only.

To avoid raising a **SqlNullValueException**, always check the **System.Data.SqlTypes.SqlBinary.IsNull** property and the **Length** property before reading **this** . The position of the byte to be retrieved.

Length

```
[C#]      public      int      Length      {get;}
```

```
[C++]      public:      __property      int      get_Length();
```

```

1  [VB]      Public      ReadOnly      Property      Length      As      Integer
2  [JScript]      public      function      get      Length()      :      int;

```

3

4 *Description*

5 Gets the length in bytes of the **System.Data.SqlTypes.SqlBinary.Value**

6 property. This property is read-only.

7 To avoid raising a **SqlNullValueException**, always check the

8 **System.Data.SqlTypes.SqlBinary.IsNull** property before reading the **Length**

9 property.

10 Value

```

11
12 [C#]      public      byte[]      Value      {get;}
13 [C++]      public:      __property      unsigned      char      get_Value();
14 [VB]      Public      ReadOnly      Property      Value      As      Byte      ()
15 [JScript]      public      function      get      Value()      :      Byte[];

```

16

17 *Description*

18 Gets the value of the **System.Data.SqlTypes.SqlBinary** structure. This

19 property is read-only.

20 To avoid raising a **SqlNullValueException**, always check the

21 **System.Data.SqlTypes.SqlBinary.IsNull** property before reading the **Value**

22 property.

23 Methods:

24 CompareTo

25

```

1
2 [C#]          public          int          CompareTo(object          value);
3 [C++]         public:         __sealed     int          CompareTo(Object*          value);
4 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
5 Integer
6 [JScript]     public     function     CompareTo(value      :   Object)      :   int;
7

```

### *Description*

Compares this **System.Data.SqlTypes.SqlBinary** object to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of this **SqlBinary** structure and the object. The object to be compared to this **SqlBinary** structure.

### **Concat**

```

13
14
15 [C#]   public   static   SqlBinary   Concat(SqlBinary   x,   SqlBinary   y);
16 [C++]  public:  static   SqlBinary   Concat(SqlBinary   x,   SqlBinary   y);
17 [VB]   Public   Shared   Function   Concat(ByVal x As SqlBinary, ByVal y As
18 SqlBinary)                                     As                               SqlBinary
19 [JScript] public static function Concat(x : SqlBinary, y : SqlBinary) : SqlBinary;
20

```

### *Description*

Concatenates two **System.Data.SqlTypes.SqlBinary** structures to create a new **SqlBinary** structure.

*Return Value:* The concatenated values of the *x* and *y* parameters. A **SqlBinary** structure. A **SqlBinary** structure.

## Equals

```
[C#]      public      override      bool      Equals(object      value);
[C++]      public:      bool      Equals(Object*      value);
[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean
[JScript] public override function Equals(value : Object) : Boolean;
```

### *Description*

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlBinary.Value** property of the **System.Data.SqlTypes.SqlBinary** object.

*Return Value:* **true** if object is an instance of **System.Data.SqlTypes.SqlBinary** and the two are equal; otherwise **false** . The object to be compared.

## Equals

```
[C#] public static new SqlBoolean Equals(SqlBinary x, SqlBinary y);
[C++] public: static SqlBoolean Equals(SqlBinary x, SqlBinary y);
[VB] Shadows Public Shared Function Equals(ByVal x As SqlBinary, ByVal y As
SqlBinary) As SqlBoolean
[JScript] public static hide function Equals(x : SqlBinary, y : SqlBinary) :
SqlBoolean; Compares two System.Data.SqlTypes.SqlBinary structures to
determine if they are equal.
```

### *Description*

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlBinary** structure. A **SqlBinary** structure.

#### GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

#### Description

Returns the hash code for this **System.Data.SqlTypes.SqlBinary** structure.

*Return Value:* A 32-bit signed integer hash code.

#### GreaterThan

[C#] public static SqlBoolean GreaterThan(SqlBinary x, SqlBinary y);

[C++] public: static SqlBoolean GreaterThan(SqlBinary x, SqlBinary y);

[VB] Public Shared Function GreaterThan(ByVal x As SqlBinary, ByVal y As SqlBinary) As SqlBoolean

```
1 [JScript] public static function GreaterThan(x : SqlBinary, y : SqlBinary) :
2 SqlBoolean;
```

#### 4 *Description*

5 Compares two **System.Data.SqlTypes.SqlBinary** structures to determine  
6 if the first is greater than the second.

7 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
8 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the  
9 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
10 instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of  
11 the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlBinary**  
12 structure. A **SqlBinary** structure.

#### 13 *GreaterThanOrEqual*

```
15 [C#] public static SqlBoolean GreaterThanOrEqual(SqlBinary x, SqlBinary y);
```

```
16 [C++] public: static SqlBoolean GreaterThanOrEqual(SqlBinary x, SqlBinary y);
```

```
17 [VB] Public Shared Function GreaterThanOrEqual(ByVal x As SqlBinary, ByVal
18 y As SqlBinary) As SqlBoolean
```

```
19 [JScript] public static function GreaterThanOrEqual(x : SqlBinary, y : SqlBinary) :
20 SqlBoolean;
```

#### 22 *Description*

23 Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if  
24 the first is greater than or equal to the second.

25 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is

**System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**

If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlBinary** structure. A **SqlBinary** structure.

**LessThan**

```
[C#] public static SqlBoolean LessThan(SqlBinary x, SqlBinary y);  
[C++] public: static SqlBoolean LessThan(SqlBinary x, SqlBinary y);  
[VB] Public Shared Function LessThan(ByVal x As SqlBinary, ByVal y As  
SqlBinary) As SqlBoolean  
[JScript] public static function LessThan(x : SqlBinary, y : SqlBinary) :  
SqlBoolean;
```

#### *Description*

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if the first is less than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlBinary** structure. A **SqlBinary** structure.

**LessThanOrEqual**

```

1
2 [C#] public static SqlBoolean LessThanOrEqual(SqlBinary x, SqlBinary y);
3 [C++] public: static SqlBoolean LessThanOrEqual(SqlBinary x, SqlBinary y);
4 [VB] Public Shared Function LessThanOrEqual(ByVal x As SqlBinary, ByVal y
5 As          SqlBinary)          As          SqlBoolean
6 [JScript] public static function LessThanOrEqual(x : SqlBinary, y : SqlBinary) :
7 SqlBoolean;
8

```

### 9 *Description*

10 Compares two **System.Data.SqlTypes.SqlBinary** structures to determine  
11 if the first is less than or equal to the second.

12 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
13 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal  
14 to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If  
15 either instance of **SqlBinary** is null, the  
16 **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be  
17 **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlBinary** structure. A **SqlBinary**  
18 structure.

### 19 *NotEquals*

```

20
21 [C#] public static SqlBoolean NotEquals(SqlBinary x, SqlBinary y);
22 [C++] public: static SqlBoolean NotEquals(SqlBinary x, SqlBinary y);
23 [VB] Public Shared Function NotEquals(ByVal x As SqlBinary, ByVal y As
24 SqlBinary)          As          SqlBoolean
25 [JScript] public static function NotEquals(x : SqlBinary, y : SqlBinary) :

```

1 SqlBoolean;

3 *Description*

4 Compares two **System.Data.SqlTypes.SqlBinary** structures to determine  
5 if they are equal.

6 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
7 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or  
8 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either  
9 instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of  
10 the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlBinary**  
11 structure. A **SqlBinary** structure.

12 op\_Addition

14 [C#] public static SqlBinary operator +(SqlBinary x, SqlBinary y);

15 [C++] public: static SqlBinary op\_Addition(SqlBinary x, SqlBinary y);

16 [VB] returnValue = SqlBinary.op\_Addition(x, y)

17 [JScript] returnValue = x + y;

19 *Description*

20 Concatenates the two **System.Data.SqlTypes.SqlBinary** parameters to  
21 create a new **SqlBinary** structure.

22 *Return Value:* The concatenated values of the *x* and *y* parameters.

23 *x* will appear first in the resulting **SqlBinary** , followed by *y* . A **SqlBinary**  
24 object. A **SqlBinary** object.

25 op\_Equality

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public static SqlBoolean operator ==(SqlBinary x, SqlBinary y);  
[C++] public: static SqlBoolean op_Equality(SqlBinary x, SqlBinary y);  
[VB]     returnValue      =      SqlBinary.op_Equality(x,      y)  
[JScript]     returnValue      =      x      ==      y;
```

*Description*

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlBinary** object. A **SqlBinary** object.

```
op_Explicit  
  
[C#] public static explicit operator byte[](SqlBinary x);  
[C++] public: static unsigned char op_Explicit();  
[VB]     returnValue      =      SqlBinary.op_Explicit(x)  
[JScript]     returnValue      =      Byte[](x);
```

*Description*

Gets the contents of the **System.Data.SqlTypes.SqlBinary.Value** property of the **System.Data.SqlTypes.SqlBinary** parameter as an array of bytes.

*Return Value:* An array of bytes.

In Visual Basic, you can use the conversions defined by the class, but you cannot override them or create your own. If Option Strict is set, you must use the to convert the **System.Data.SqlTypes.SqlBinary** to a binary object. A **System.Data.SqlTypes.SqlBinary**.

op\_Explicit

[C#] public static explicit operator SqlBinary(SqlGuid x);

[C++] public: static SqlBinary op\_Explicit(SqlGuid x);

[VB] returnValue = SqlBinary.op\_Explicit(x)

[JScript] returnValue = SqlBinary(x);

### Description

Converts a **System.Data.SqlTypes.SqlGuid** structure to a **System.Data.SqlTypes.SqlBinary** structure.

*Return Value:* A **SqlBinary** structure.

In Visual Basic, you can use the conversions defined by the class, but you cannot override them or create your own. If Option Strict is set, you must use the to convert the **System.Data.SqlTypes.SqlGuid** to a **System.Data.SqlTypes.SqlBinary**. The **SqlGuid** structure to be converted.

op\_GreaterThan

[C#] public static SqlBoolean operator >(SqlBinary x, SqlBinary y);

```

1  [C++] public: static SqlBoolean op_GreaterThan(SqlBinary x, SqlBinary y);
2  [VB]      returnValue = SqlBinary.op_GreaterThan(x, y)
3  [JScript]      returnValue = x > y;

```

### Description

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if the first is greater than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlBinary** object. A **SqlBinary** object.

## op GreaterThanOrEqualTo

```
[C#] public static SqlBoolean operator >=(SqlBinary x, SqlBinary y);
[C++] public: static SqlBoolean op_GreaterThanOrEqual(SqlBinary x, SqlBinary
y);
```

[VB]	returnValue	=	SqlBinary.op_GreaterThanOrEqual(x,	y)
[JScript]	returnValue	=	x	>= y;

### Description

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if the first is greater than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is



## Description

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if they are equal.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlBinary** object. A **SqlBinary** object.

op\_LessThan

[C#] public static SqlBoolean operator

[C++] public: static SqlBoolean op\_LessThan(SqlBinary x, SqlBinary y);

[VB] returnValue = SqlBinary.op\_LessThan(x, y)

[JScript] returnValue = x < y;

## Description

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if the first is less than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of

the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlBinary** object. A **SqlBinary** object.

**op\_LessThanOrEqual**

[C#] public static SqlBoolean operator <=(SqlBinary x, SqlBinary y);  
 [C++] public: static SqlBoolean op\_LessThanOrEqual(SqlBinary x, SqlBinary y);  
 [VB] returnValue = SqlBinary.op\_LessThanOrEqual(x, y)  
 [JScript] returnValue = x <= y;

*Description*

Compares two **System.Data.SqlTypes.SqlBinary** structures to determine if the first is less than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **SqlBinary** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlBinary** object. A **SqlBinary** object.

**ToSqlGuid**

[C#] public SqlGuid ToSqlGuid();  
 [C++] public: SqlGuid ToSqlGuid();  
 [VB] Public Function ToSqlGuid() As SqlGuid  
 [JScript] public function ToSqlGuid() : SqlGuid;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Converts this instance of **System.Data.SqlTypes.SqlBinary** to **System.Data.SqlTypes.SqlGuid** .

**ToString**

```
[C#]          public          override          string          ToString();
[C++]          public:          String*          ToString();
[VB]  Overrides  Public  Function  ToString()  As  String
[JScript] public override function ToString() : String; Converts a
System.Data.SqlTypes.SqlBinary          to          a          string.
```

*Description*

Converts this **System.Data.SqlTypes.SqlBinary** object to a string.  
*Return Value:* A string containing the **System.Data.SqlTypes.SqlBinary.Value** of the **SqlBinary** . If the **Value** is null the string will contain "null".

**SqlBoolean** structure (System.Data.SqlTypes)  
**ToString**

*Description*

Represents an integer value that is either 1 or 0 to be stored in or retrieved from a database.

1 Any non-zero value is interpreted as 1.

2

3 *Description*

4 Represents a boolean stored in or retrieved from a database.

5 The key difference between a **SqlBoolean** structure and a standard boolean  
6 value is that, where a standard boolean has two possible values, **true** and **false** , a

7 **SqlBoolean** structure has three possible values,

8 **System.Data.SqlTypes.SqlBoolean.True** ,

9 **System.Data.SqlTypes.SqlBoolean.False** , or

10 **System.Data.SqlTypes.SqlBoolean.Null** .

11 ToString

12

13 [C#] public static readonly SqlBoolean False;

14 [C++] public: static SqlBoolean False;

15 [VB] Public Shared ReadOnly False As SqlBoolean

16 [JScript] public static var False : SqlBoolean;

17

18 *Description*

19 Represents a false value that can be assigned to the  
20 **System.Data.SqlTypes.SqlBoolean.Value** property of an instance of the  
21 **System.Data.SqlTypes.SqlBoolean** structure.

22 The **System.Data.SqlTypes.SqlBoolean.False** field is a constant for the  
23 **System.Data.SqlTypes.SqlBoolean** structure.

24 ToString

25

TOP SECRET

```

1
2 [C#]      public      static      readonly      SqlBoolean      Null;
3 [C++]      public:      static      SqlBoolean      Null;
4 [VB]      Public      Shared      ReadOnly      Null      As      SqlBoolean
5 [JScript]      public      static      var      Null      :      SqlBoolean;
6

```

### Description

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlBoolean.Value** property of an instance of the **System.Data.SqlTypes.SqlBoolean** structure.

The **System.Data.SqlTypes.SqlBoolean.Null** field is a constant for the **System.Data.SqlTypes.SqlBoolean** structure.

### Description

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlBoolean.ByteValue** property or the **System.Data.SqlTypes.SqlBoolean.BoolValue** of an instance of the **System.Data.SqlTypes.SqlBoolean** structure.

**System.Data.SqlTypes.SqlBoolean.Null** functions as a constant for the **System.Data.SqlTypes.SqlBoolean** structure.

### ToString

```

22
23 [C#]      public      static      readonly      SqlBoolean      One;
24 [C++]      public:      static      SqlBoolean      One;
25 [VB]      Public      Shared      ReadOnly      One      As      SqlBoolean

```

1 [JScript] public static var One : SqlBoolean;

2

3 *Description*

4 Represents a one value that can be assigned to the  
5 **System.Data.SqlTypes.SqlBoolean.ByteValue** property of an instance of the  
6 **System.Data.SqlTypes.SqlBoolean** structure.

7 The **System.Data.SqlTypes.SqlBoolean.One** field is a constant for the  
8 **System.Data.SqlTypes.SqlBoolean** structure.

9 ToString

10

11 [C#] public static readonly SqlBoolean True;

12 [C++] public: static SqlBoolean True;

13 [VB] Public Shared ReadOnly True As SqlBoolean

14 [JScript] public static var True : SqlBoolean;

15

16 *Description*

17 Represents a true value that can be assigned to the  
18 **System.Data.SqlTypes.SqlBoolean.Value** property of an instance of the  
19 **System.Data.SqlTypes.SqlBoolean** structure.

20 The **System.Data.SqlTypes.SqlBoolean.True** field is a constant for the  
21 **System.Data.SqlTypes.SqlBoolean** structure.

22 ToString

23

24 [C#] public static readonly SqlBoolean Zero;

25 [C++] public: static SqlBoolean Zero;

1 [VB] Public Shared ReadOnly Zero As SqlBoolean  
 2 [JScript] public static var Zero : SqlBoolean;

4 *Description*

5 Represents a zero value that can be assigned to the  
 6 **System.Data.SqlTypes.SqlBoolean.ByteValue** property of an instance of the  
 7 **System.Data.SqlTypes.SqlBoolean** structure.

8 The **System.Data.SqlTypes.SqlBoolean.Zero** field is a constant for the  
 9 **System.Data.SqlTypes.SqlBoolean** structure.

10 SqlBoolean

11 *Example Syntax:*

12 ToString

14 [C#] public SqlBoolean(bool value);

15 [C++] public: SqlBoolean(bool value);

16 [VB] Public Sub New(ByVal value As Boolean)

17 [JScript] public function SqlBoolean(value : Boolean); Initializes a new instance  
 18 of the **System.Data.SqlTypes.SqlBoolean** structure.

20 *Description*

21 Initializes a new instance of the **System.Data.SqlTypes.SqlBoolean**  
 22 structure with a boolean value to be stored.

24 *Description*

25

1        Initializes a new instance of the **System.Data.SqlTypes.SqlBoolean**  
 2 structure using the supplied boolean value. The boolean value to be stored.

3        **SqlBoolean**

4        *Example Syntax:*

5        **ToString**

6  
 7        [C#]                    public                    **SqlBoolean**(int                    value);  
 8        [C++]                    public:                    **SqlBoolean**(int                    value);  
 9        [VB]        Public        Sub        New(ByVal        value        As        Integer)  
 10        [JScript]        public        function        **SqlBoolean**(value        :        int);

11  
 12        *Description*

13        Initializes a new instance of the **System.Data.SqlTypes.SqlBoolean**  
 14 structure using the specified **integer** value. The integer whose value is to be used  
 15 for the new **SqlBoolean** structure.

16        **ByteValue**

17        **ToString**

18  
 19        [C#]                    public                    byte                    **ByteValue**                    {get;}  
 20        [C++]        public:        \_\_property        unsigned        char        get\_ByteValue();  
 21        [VB]        Public        ReadOnly        Property        **ByteValue**        As        Byte  
 22        [JScript]        public        function        get        **ByteValue**()        :        Byte;

23  
 24        *Description*

25

1 Gets the value of the **System.Data.SqlTypes.SqlBoolean** structure as a  
2 byte.

3 The byte value will be either 0 or 1.

4 **IsFalse**

5 **ToString**

6  
7 [C#] public bool **IsFalse** {get;}

8 [C++] public: \_\_property bool get\_IsFalse();

9 [VB] Public ReadOnly Property **IsFalse** As Boolean

10 [JScript] public function get **IsFalse()** : Boolean;

11  
12 *Description*

13 Indicates whether the current **System.Data.SqlTypes.SqlBoolean.Value** is  
14 **System.Data.SqlTypes.SqlBoolean.False** .

15 If the **System.Data.SqlTypes.SqlBoolean.Value** is  
16 **System.Data.SqlTypes.SqlBoolean.Null** , this property still will be **false** .

17 **IsNull**

18 **ToString**

19  
20 [C#] public bool **IsNull** {get;}

21 [C++] public: \_\_property bool get\_IsNull();

22 [VB] Public ReadOnly Property **IsNull** As Boolean

23 [JScript] public function get **IsNull()** : Boolean;

24  
25 *Description*

Indicates whether or not the value of the  
**System.Data.SqlTypes.SqlBoolean** structure is null.

#### *Description*

Indicates whether the current **System.Data.SqlTypes.SqlBoolean.Value** is  
**System.Data.SqlTypes.SqlBoolean.Null** .

IsTrue

ToString

[C#]            public            bool            IsTrue            {get;}

[C++]           public:           \_\_property           bool           get\_IsTrue();

[VB]           Public           ReadOnly           Property           IsTrue           As           Boolean

[JScript]       public           function           get           IsTrue()           :           Boolean;

#### *Description*

Indicates whether the current **System.Data.SqlTypes.SqlBoolean.Value** is  
**System.Data.SqlTypes.SqlBoolean.True** .

If the **System.Data.SqlTypes.SqlBoolean.Value** is  
**System.Data.SqlTypes.SqlBoolean.Null** , this property still will be **false** .

Value

ToString

[C#]            public            bool            Value            {get;}

[C++]           public:           \_\_property           bool           get\_Value();

[VB]           Public           ReadOnly           Property           Value           As           Boolean

1 [JScript] public function get Value() : Boolean;

2

3 *Description*

4 Gets the **System.Data.SqlTypes.SqlBoolean** structure's value. This  
5 property is read-only.

6 And

7

8 [C#] public static SqlBoolean And(SqlBoolean x, SqlBoolean y);

9 [C++] public: static SqlBoolean And(SqlBoolean x, SqlBoolean y);

10 [VB] Public Shared Function And(ByVal x As SqlBoolean, ByVal y As  
11 SqlBoolean) As SqlBoolean

12 [JScript] public static function And(x : SqlBoolean, y : SqlBoolean) : SqlBoolean;

13

14 *Description*

15 Computes the bitwise AND of two specified  
16 **System.Data.SqlTypes.SqlBoolean** structures.

17 *Return Value:* The result of the logical AND operation. A **SqlBoolean** structure. A  
18 **SqlBoolean** structure.

19 CompareTo

20

21 [C#] public int CompareTo(object value);

22 [C++] public: \_\_sealed int CompareTo(Object\* value);

23 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As  
24 Integer

25 [JScript] public function CompareTo(value : Object) : int;

## Description

Compares this **System.Data.SqlTypes.SqlBoolean** structure to a specified object and returns an indication of their relative values.

**Return Value:** A signed number indicating the relative values of the instance and value.

Any instance of **SqlBoolean**, regardless of its value, is considered greater than a null reference ( **Nothing** ). An object to compare, or a null reference (**Nothing** in Visual Basic).

## Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object\* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean;

## Description

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlBoolean**.

**Return Value:** **true** if object is an instance of **System.Data.SqlTypes.SqlBoolean** and the two are equal; otherwise **false**. The object to be compared.

## Equals

[C#] public static new SqlBoolean Equals(SqlBoolean x, SqlBoolean y);

[C++] public: static SqlBoolean Equals(SqlBoolean x, SqlBoolean y);

1 [VB] Shadows Public Shared Function Equals(ByVal x As SqlBoolean, ByVal y  
2 As SqlBoolean) As SqlBoolean  
3 [JScript] public static hide function Equals(x : SqlBoolean, y : SqlBoolean) :  
4 SqlBoolean; Compares two **System.Data.SqlTypes.SqlBoolean** structures to  
5 determine if they are equal.

6  
7 *Description*

8 Compares two **System.Data.SqlTypes.SqlBoolean** structures to determine  
9 if they are equal.

10 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
11 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or  
12 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If  
13 either instance of **System.Data.SqlTypes.SqlBoolean** is null, the  
14 **System.Data.SqlTypes.SqlBoolean.Value** of the  
15 **System.Data.SqlTypes.SqlBoolean** will be  
16 **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlBoolean** structure. A  
17 **SqlBoolean** structure.

18 *GetHashCode*

19  
20 [C#] public override int GetHashCode();  
21 [C++] public: int GetHashCode();  
22 [VB] Overrides Public Function GetHashCode() As Integer  
23 [JScript] public override function GetHashCode() : int;

24  
25 *Description*

1 Returns the hash code for this instance.

2 *Return Value:* A 32-bit signed integer hash code.

3 NotEquals

4  
5 [C#] public static SqlBoolean NotEquals(SqlBoolean x, SqlBoolean y);

6 [C++] public: static SqlBoolean NotEquals(SqlBoolean x, SqlBoolean y);

7 [VB] Public Shared Function NotEquals(ByVal x As SqlBoolean, ByVal y As  
8 SqlBoolean) As SqlBoolean

9 [JScript] public static function NotEquals(x : SqlBoolean, y : SqlBoolean) :  
10 SqlBoolean;

11  
12 *Description*

13 Compares two instances of **System.Data.SqlTypes.SqlBoolean** for  
14 equality.

15 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
16 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or  
17 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either  
18 instance of **System.Data.SqlTypes.SqlBoolean** is null, the  
19 **System.Data.SqlTypes.SqlBoolean.Value** of the  
20 **System.Data.SqlTypes.SqlBoolean** will be  
21 **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlBoolean** structure. A  
22 **SqlBoolean** structure.

23 OnesComplement

24  
25 [C#] public static SqlBoolean OnesComplement(SqlBoolean x);

```

1  [C++] public: static SqlBoolean OnesComplement(SqlBoolean x);
2  [VB] Public Shared Function OnesComplement(ByVal x As SqlBoolean) As
3  SqlBoolean
4  [JScript] public static function OnesComplement(x : SqlBoolean) : SqlBoolean;

```

#### 6 *Description*

7 Performs a one's complement operation on the supplied  
8 **System.Data.SqlTypes.SqlBoolean** structures.

9 *Return Value:* The one's complement of the supplied  
10 **System.Data.SqlTypes.SqlBoolean** . A **SqlBoolean** structure.

11 op\_BitwiseAnd

```

12
13 [C#] public static SqlBoolean operator &(amp;SqlBoolean x, SqlBoolean y);
14 [C++] public: static SqlBoolean op_BitwiseAnd(SqlBoolean x, SqlBoolean y);
15 [VB]     returnValue     =     SqlBoolean.op_BitwiseAnd(x,     y)
16 [JScript]     returnValue     =     x     &     y;

```

#### 18 *Description*

19 Performs a bitwise AND operation on two  
20 **System.Data.SqlTypes.SqlBoolean** structures.

21 *Return Value:* A **SqlBoolean** structure that is the result of the bitwise AND  
22 operation.

#### 24 *Description*

25

3 *Return Value:* The results of the logical AND operation. The **SqlBoolean** . The  
4 **SqlBoolean** .

s	op_BitwiseOr
---	--------------

```
7 | [C#] public static SqlBoolean operator |(SqlBoolean x, SqlBoolean y);
```

```
8 [C++] public: static SqlBoolean op_BitwiseOr(SqlBoolean x, SqlBoolean y);
```

```
9  [VB]    returnValue    =    SqlBoolean.op_BooleanOr(x,    y)
```

10	[JScript]	returnValue	=	x		y;
----	-----------	-------------	---	---	--	----

12	Description
----	-------------

13	Computes the bitwise OR of its operands.
----	--

14 **Return Value:** The results of the logical OR operation.

16	Description
----	-------------

17	Performs a bitwise OR operation on the two specified
----	--

18 **System.Data.SqlTypes.SqlBoolean** structures.

19 **Return Value:** A new **SqlBoolean** whose **Value** is the result of the bitwise OR  
20 operation. A **System.Data.SqlTypes.SqlBoolean** structure. A  
21 **System.Data.SqlTypes.SqlBoolean** structure.

22 op\_Equality

```
24 [C#] public static SqlBoolean operator ==(SqlBoolean x, SqlBoolean y);
```

```
25 [C++] public: static SqlBoolean op_Equality(SqlBoolean x, SqlBoolean y);
```

1 [VB]           returnValue           =           SqlBoolean.op\_Equality(x,           y)  
2 [JScript]           returnValue           =           x           ==           y;

3

4 *Description*

5       Compares two instances of **System.Data.SqlTypes.SqlBoolean** for  
6 equality.

7 *Return Value:*   A **System.Data.SqlTypes.SqlBoolean** that is  
8 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or  
9 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If  
10 either instance of **System.Data.SqlTypes.SqlBoolean** is null, the  
11 **System.Data.SqlTypes.SqlBoolean.Value** of the  
12 **System.Data.SqlTypes.SqlBoolean** will be  
13 **System.Data.SqlTypes.SqlBoolean.Null**. A  
14 **System.Data.SqlTypes.SqlBoolean**. A **System.Data.SqlTypes.SqlBoolean**.

15       op\_ExclusiveOr

16

17 [C#] public static SqlBoolean operator ^(SqlBoolean x, SqlBoolean y);

18 [C++] public: static SqlBoolean op\_ExclusiveOr(SqlBoolean x, SqlBoolean y);

19 [VB]           returnValue           =           SqlBoolean.op\_ExclusiveOr(x,           y)

20 [JScript]           returnValue           =           x           ^           y;

21

22 *Description*

23       Performs a bitwise exclusive-OR operation on the supplied parameters.

24 *Return Value:*   The results of the logical XOR operation. A

25

**System.Data.SqlTypes.SqlBoolean** structure. A

**System.Data.SqlTypes.SqlBoolean** structure.

op\_Explicit

[C#] public static explicit operator bool(SqlBoolean x);  
 [C++] public: static bool op\_Explicit();  
 [VB] returnValue = SqlBoolean.op\_Explicit(x)  
 [JScript] returnValue = Boolean(x);

*Description*

Converts a **System.Data.SqlTypes.SqlBoolean** to a boolean.

*Return Value:* A boolean set to the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean**. A **SqlBoolean** to convert.

op\_Explicit

[C#] public static explicit operator SqlBoolean(SqlByte x);  
 [C++] public: static SqlBoolean op\_Explicit(SqlByte x);  
 [VB] returnValue = SqlBoolean.op\_Explicit(x)  
 [JScript] returnValue = SqlBoolean(x);

*Description*

Converts the **System.Data.SqlTypes.SqlByte** parameter to a **System.Data.SqlTypes.SqlBoolean** structure.

*Return Value:* A new **System.Data.SqlTypes.SqlBoolean** structure whose value equals the **System.Data.SqlTypes.SqlByte.Value** of the

**System.Data.SqlTypes.SqlByte** parameter. A **System.Data.SqlTypes.SqlByte** to be converted to a **System.Data.SqlTypes.SqlBoolean** structure.

op\_Explicit

[C#] public static explicit operator SqlBoolean(SqlDecimal x);

[C++] public: static SqlBoolean op\_Explicit(SqlDecimal x);

[VB] returnValue = SqlBoolean.op\_Explicit(x)

[JScript] returnValue = SqlBoolean(x);

#### *Description*

Converts the **System.Data.SqlTypes.SqlDecimal** parameter to a **System.Data.SqlTypes.SqlBoolean** structure.

*Return Value:* A new **System.Data.SqlTypes.SqlByte** structure whose value equals the **System.Data.SqlTypes.SqlDecimal.Value** property of the **System.Data.SqlTypes.SqlDecimal** parameter. A **System.Data.SqlTypes.SqlDecimal** to be converted to a **System.Data.SqlTypes.SqlBoolean** structure.

op\_Explicit

[C#] public static explicit operator SqlBoolean(SqlDouble x);

[C++] public: static SqlBoolean op\_Explicit(SqlDouble x);

[VB] returnValue = SqlBoolean.op\_Explicit(x)

[JScript] returnValue = SqlBoolean(x);

#### *Description*

Converts the **System.Data.SqlTypes.SqlDouble** parameter to a **System.Data.SqlTypes.SqlBoolean** structure.

*Return Value:* A new **SqlBoolean** structure whose value equals the **System.Data.SqlTypes.SqlDouble.Value** property of the **SqlDouble** parameter.

A **SqlDouble** to be converted to a **SqlBoolean** structure.

op\_Explicit

[C#] public static explicit operator SqlBoolean(SqlInt16 x);

[C++] public: static SqlBoolean op\_Explicit(SqlInt16 x);

[VB] returnValue = SqlBoolean.op\_Explicit(x)

[JScript] returnValue = SqlBoolean(x);

### Description

Converts the **System.Data.SqlTypes.SqlInt16** parameter to a **System.Data.SqlTypes.SqlBoolean** structure.

*Return Value:* A new **SqlBoolean** structure whose value equals the **System.Data.SqlTypes.SqlInt16.Value** property of the **SqlInt16** parameter. A

**SqlInt16** to be converted to a **SqlBoolean** structure.

op\_Explicit

[C#] public static explicit operator SqlBoolean(SqlInt32 x);

[C++] public: static SqlBoolean op\_Explicit(SqlInt32 x);

[VB] returnValue = SqlBoolean.op\_Explicit(x)

[JScript] returnValue = SqlBoolean(x);

## Description

Converts the **System.Data.SqlTypes.SqlInt32** parameter to a **System.Data.SqlTypes.SqlBoolean** structure.

*Return Value:* A new **SqlBoolean** structure whose value equals the **System.Data.SqlTypes.SqlInt32.Value** property of the **SqlInt32** parameter. A **SqlInt32** to be converted to a **SqlBoolean** structure.

op\_Explicit

```
[C#] public static explicit operator SqlBoolean(SqlInt64 x);
```

```
[C++] public: static SqlBoolean op_Explicit(SqlInt64 x);
```

```
[VB] returnValue = SqlBoolean.op_Explicit(x)
```

```
[JScript] returnValue = SqlBoolean(x);
```

## Description

Converts the **System.Data.SqlTypes.SqlInt64** parameter to a **System.Data.SqlTypes.SqlBoolean** structure.

*Return Value:* A new **SqlBoolean** structure whose value equals the **System.Data.SqlTypes.SqlInt64.Value** property of the **SqlInt64** parameter. A **SqlInt64** to be converted to a **SqlBoolean** structure.

op\_Explicit

```
[C#] public static explicit operator SqlBoolean(SqlMoney x);
```

```
[C++] public: static SqlBoolean op_Explicit(SqlMoney x);
```

```
[VB] returnValue = SqlBoolean.op_Explicit(x)
```



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]    public    static    explicit    operator    SqlBoolean(SqlString    x);
[C++]    public:    static    SqlBoolean    op_Explicit(SqlString    x);
[VB]        returnValue    =    SqlBoolean.op_Explicit(x)
[JScript]        returnValue    =    SqlBoolean(x);
```

*Description*

Converts the **System.Data.SqlTypes.SqlString** parameter to a **System.Data.SqlTypes.SqlBoolean** structure.

*Return Value:* A new **System.Data.SqlTypes.SqlByte** structure whose value equals the **System.Data.SqlTypes.SqlBoolean.Value** property of the **System.Data.SqlTypes.SqlBoolean** parameter. A **System.Data.SqlTypes.SqlString** to be converted to a **System.Data.SqlTypes.SqlBoolean** structure.

op\_False

```
[C#]    public    static    bool    operator    false(SqlBoolean    x);
[C++]    public:    static    bool    op_False(SqlBoolean    x);
[VB]        returnValue    =    SqlBoolean.op_False(x)
```

*Description*

The **false** operator can be used to test the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** to determine whether it is false.

*Return Value:* Returns **true** if the supplied parameter is **SqlBoolean** is false, **false** otherwise. The **System.Data.SqlTypes.SqlBoolean** structure to be tested.

op\_Implicit

[C#] public static implicit operator SqlBoolean(bool x);

[C++] public: static SqlBoolean op\_Implicit(bool x);

[VB] returnValue = SqlBoolean.op\_Implicit(x)

[JScript] returnValue = x;

#### *Description*

Converts the supplied byte value to a **System.Data.SqlTypes.SqlBoolean**.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** value containing 0 or 1.

#### *Description*

Converts a boolean to a **System.Data.SqlTypes.SqlBoolean**.

*Return Value:* A **SqlBoolean** with a **System.Data.SqlTypes.SqlBoolean.Value** equivalent to the parameter. A byte value to be converted to **System.Data.SqlTypes.SqlBoolean**.

op\_Inequality

[C#] public static SqlBoolean operator !=(SqlBoolean x, SqlBoolean y);

[C++] public: static SqlBoolean op\_Inequality(SqlBoolean x, SqlBoolean y);

[VB] returnValue = SqlBoolean.op\_Inequality(x, y)

[JScript] returnValue = x != y;

## Description

Compares two instances of **System.Data.SqlTypes.SqlBoolean** for equality.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **System.Data.SqlTypes.SqlBoolean** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlBoolean**. A **System.Data.SqlTypes.SqlBoolean**.

op\_LogicalNot

[C#] public static SqlBoolean operator !(SqlBoolean x);

[C++] public: static SqlBoolean op\_LogicalNot(SqlBoolean x);

[VB] returnValue = SqlBoolean.op\_LogicalNot(x)

[JScript] returnValue = !x;

## Description

Performs a NOT operation on a **System.Data.SqlTypes.SqlBoolean**.

**Return Value:** A **SqlBoolean** with the **System.Data.SqlTypes.SqlBoolean.ValueSystem.Data.SqlTypes.SqlBoolean.True** if argument was true, **System.Data.SqlTypes.SqlBoolean.Null** if argument

was null, and **System.Data.SqlTypes.SqlBoolean.False** otherwise. The **SqlBoolean** on which the NOT operation will be performed.

**op\_OnesComplement**

[C#] public static SqlBoolean operator ~(SqlBoolean x);

[C++] public: static SqlBoolean op\_OnesComplement(SqlBoolean x);

[VB] returnValue = SqlBoolean.op\_OnesComplement(x)

[JScript] returnValue = ~x;

#### *Description*

Performs a one's complement operation on the supplied **System.Data.SqlTypes.SqlBoolean** structures.

*Return Value:* The one's complement of the supplied **System.Data.SqlTypes.SqlBoolean**. A **System.Data.SqlTypes.SqlBoolean** structure.

**op\_True**

[C#] public static bool operator true(SqlBoolean x);

[C++] public: static bool op\_True(SqlBoolean x);

[VB] returnValue = SqlBoolean.op\_True(x)

#### *Description*

The **true** operator can be used to test the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** to determine whether it is true.

*Return Value:* Returns **true** if the supplied parameter is **SqlBoolean** is true, **false** otherwise. The **SqlBoolean** structure to be tested.

Or

```
[C#]    public static SqlBoolean Or(SqlBoolean x, SqlBoolean y);
[C++]   public: static SqlBoolean Or(SqlBoolean x, SqlBoolean y);
[VB]    Public Shared Function Or(ByVal x As SqlBoolean, ByVal y As
SqlBoolean)
                                As                               SqlBoolean
[JScript] public static function Or(x : SqlBoolean, y : SqlBoolean) : SqlBoolean;
```

#### *Description*

Performs a bitwise OR operation on the two specified **System.Data.SqlTypes.SqlBoolean** structures.

*Return Value:* A new **SqlBoolean** structure whose Value is the result of the bitwise OR operation. A **SqlBoolean** structure. A **SqlBoolean** structure.

Parse

```
[C#]    public static SqlBoolean Parse(string s);
[C++]   public: static SqlBoolean Parse(String* s);
[VB]    Public Shared Function Parse(ByVal s As String) As SqlBoolean
[JScript] public static function Parse(s : String) : SqlBoolean;
```

#### *Description*

[ . ] [ . ]

ToSqlByte

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]          public          SqlByte          ToSqlByte();
[C++]         public:         SqlByte          ToSqlByte();
[VB]          Public          Function          ToSqlByte()    As          SqlByte
[JScript]     public          function          ToSqlByte()    :          SqlByte;
```

*Description*

Converts this **System.Data.SqlTypes.SqlBoolean** structure to **System.Data.SqlTypes.SqlByte**.

*Return Value:* A **SqlByte** structure whose **Value** equals the **Value** of this **SqlBoolean** structure. If the **SqlBoolean** structure's **Value** is **true**, then the **SqlByte** structure's **Value** will be 1, otherwise the **SqlByte** structure's **Value** will be 0.

**ToSqlDecimal**

```
[C#]          public          SqlDecimal          ToSqlDecimal();
[C++]         public:         SqlDecimal          ToSqlDecimal();
[VB]          Public          Function          ToSqlDecimal()  As          SqlDecimal
[JScript]     public          function          ToSqlDecimal()  :          SqlDecimal;
```

*Description*

Converts this **System.Data.SqlTypes.SqlBoolean** structure to **System.Data.SqlTypes.SqlDecimal**.

*Return Value:* A new **SqlDecimal** structure whose **Value** equals 1 if the

**SqlBoolean** structure's **Value** was **true** , otherwise the **Value** of the new **SqlDecimal** structure is 0.

**ToSqlDouble**

[C#]	public	SqlDouble	ToSqlDouble();
[C++]	public:	SqlDouble	ToSqlDouble();
[VB]	Public	Function	ToSqlDouble() As SqlDouble
[JScript]	public	function	ToSqlDouble() : SqlDouble;

*Description*

Converts this **System.Data.SqlTypes.SqlBoolean** structure to **System.Data.SqlTypes.SqlDouble**

*Return Value:* A new **SqlDouble** structure whose **Value** equals 1 if the **SqlBoolean** structure's **Value** was **true** , otherwise the **Value** of the new **SqlDouble** structure is 0.

**ToSqlInt16**

[C#]	public	SqlInt16	ToSqlInt16();
[C++]	public:	SqlInt16	ToSqlInt16();
[VB]	Public	Function	ToSqlInt16() As SqlInt16
[JScript]	public	function	ToSqlInt16() : SqlInt16;

*Description*

Converts this **System.Data.SqlTypes.SqlBoolean** structure to **System.Data.SqlTypes.SqlInt16**

1 *Return Value:* A new **SqlInt16** structure whose **Value** equals 1 if the **SqlBoolean**  
 2 structure's **Value** was **true** , otherwise the **Value** of the new **SqlInt16** structure is  
 3 0.

4       **ToSqlInt32**

6 [C#]               public               SqlInt32               ToSqlInt32();  
 7 [C++]             public:             SqlInt32             ToSqlInt32();  
 8 [VB]       Public       Function       ToSqlInt32()       As       SqlInt32  
 9 [JScript]       public       function       ToSqlInt32()       :       SqlInt32;

11 *Description*

12       Converts   this   **System.Data.SqlTypes.SqlBoolean**   structure   to  
 13 **System.Data.SqlTypes.SqlInt32**

14 *Return Value:* A new **SqlInt32** structure whose **Value** equals 1 if the **SqlBoolean**  
 15 structure's **Value** was **true** , otherwise the **Value** of the new **SqlInt32** structure is  
 16 0.

17       **ToSqlInt64**

19 [C#]               public               SqlInt64               ToSqlInt64();  
 20 [C++]             public:             SqlInt64             ToSqlInt64();  
 21 [VB]       Public       Function       ToSqlInt64()       As       SqlInt64  
 22 [JScript]       public       function       ToSqlInt64()       :       SqlInt64;

24 *Description*

25

1        Converts    this    **System.Data.SqlTypes.SqlBoolean**    structure    to  
2    **System.Data.SqlTypes.SqlInt64**    .

3    *Return Value:* A new **SqlInt64** structure whose **Value** equals 1 if the **SqlBoolean**  
4    structure's **Value** was **true** , otherwise the **Value** of the new **SqlInt64** structure is  
5    0.

6        **ToSqlMoney**

7  
8    [C#]                public                **SqlMoney**                **ToSqlMoney();**  
9    [C++]                public:                **SqlMoney**                **ToSqlMoney();**  
10    [VB]        Public        Function        **ToSqlMoney()**        As        **SqlMoney**  
11    [JScript]        public        function        **ToSqlMoney()**        :        **SqlMoney;**

12  
13    *Description*

14        Converts    this    **System.Data.SqlTypes.SqlBoolean**    structure    to  
15    **System.Data.SqlTypes.SqlMoney** .

16        **ToSqlSingle**

17  
18    [C#]                public                **SqlSingle**                **ToSqlSingle();**  
19    [C++]                public:                **SqlSingle**                **ToSqlSingle();**  
20    [VB]        Public        Function        **ToSqlSingle()**        As        **SqlSingle**  
21    [JScript]        public        function        **ToSqlSingle()**        :        **SqlSingle;**

22  
23    *Description*

24        Converts    this    **System.Data.SqlTypes.SqlBoolean**    structure    to  
25    **System.Data.SqlTypes.SqlSingle** .

1 *Return Value:* A new **SqlSingle** structure whose **Value** equals 1 if the **SqlBoolean**  
2 structure's **Value** was **true** , otherwise the **Value** of the new **SqlSingle** structure is  
3 0.

4       **ToSqlString**

6 [C#]               public               SqlString               ToSqlString();  
7 [C++]              public:              SqlString              ToSqlString();  
8 [VB]       Public       Function       ToSqlString()       As       SqlString  
9 [JScript]       public       function       ToSqlString()       :       SqlString;

11 *Description*

12       Converts    this    **System.Data.SqlTypes.SqlBoolean**    structure    to  
13 **System.Data.SqlTypes.SqlString**

14 *Return Value:* A new **SqlString** structure whose **Value** equals 1 if the **SqlBoolean**  
15 structure's **Value** was **true** , otherwise the **Value** of the new **SqlDouble** structure is  
16 0.

17       **ToString**

19 [C#]               public               override               string               ToString();  
20 [C++]              public:              String\*              ToString();  
21 [VB]       Overrides       Public       Function       ToString()       As       String  
22 [JScript]       public       override       function       ToString()       :       String;

24 *Description*

25

Converts the current **System.Data.SqlTypes.SqlBoolean.Value** to a string.

*Return Value:* A string containing "true" if **true** , "null" if null, otherwise "false".

#### *Description*

Converts this **System.Data.SqlTypes.SqlBoolean** structure to a string.

*Return Value:* A string containing the value of the **System.Data.SqlTypes.SqlBoolean** . If the value is null the string will contain "null".

#### **Xor**

```
[C#] public static SqlBoolean Xor(SqlBoolean x, SqlBoolean y);
```

```
[C++] public: static SqlBoolean Xor(SqlBoolean x, SqlBoolean y);
```

```
[VB] Public Shared Function Xor(ByVal x As SqlBoolean, ByVal y As  
SqlBoolean) As SqlBoolean
```

```
[JScript] public static function Xor(x : SqlBoolean, y : SqlBoolean) : SqlBoolean;
```

#### *Description*

Performs a bitwise exclusive-OR operation on the supplied parameters.

*Return Value:* The results of the logical XOR operation. A **SqlBoolean** structure.

**SqlByte** structure (System.Data.SqlTypes)

#### **Xor**

#### *Description*

Represents an 8-bit unsigned integer, in the range of 0 through 255, to be stored in or retrieved from a database.

Xor

[C#] public static readonly SqlByte MaxValue;

[C++] public: static SqlByte MaxValue;

[VB] Public Shared ReadOnly MaxValue As SqlByte

[JScript] public static var MaxValue : SqlByte;

#### Description

A constant representing the largest possible value of a **System.Data.SqlTypes.SqlByte**.

The value of this constant is 255 or, hexadecimal 0xFF.

Xor

[C#] public static readonly SqlByte MinValue;

[C++] public: static SqlByte MinValue;

[VB] Public Shared ReadOnly MinValue As SqlByte

[JScript] public static var MinValue : SqlByte;

#### Description

A constant representing the smallest possible value of a **System.Data.SqlTypes.SqlByte**.

The value of this constant is 0.

Xor

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      static      readonly      SqlByte      Null;
[C++]      public:      static      SqlByte      Null;
[VB]      Public      Shared      ReadOnly      Null      As      SqlByte
[JScript]      public      static      var      Null      :      SqlByte;
```

*Description*

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlByte.Value** property of an instance of the **System.Data.SqlTypes.SqlByte** structure.

**Null** functions as a constant for the **SqlByte** structure.

Xor

```
[C#]      public      static      readonly      SqlByte      Zero;
[C++]      public:      static      SqlByte      Zero;
[VB]      Public      Shared      ReadOnly      Zero      As      SqlByte
[JScript]      public      static      var      Zero      :      SqlByte;
```

*Description*

Represents a zero value that can be assigned to the **System.Data.SqlTypes.SqlByte.Value** property of an instance of the **System.Data.SqlTypes.SqlByte** structure.

The **System.Data.SqlTypes.SqlByte.Zero** field is a constant for the **System.Data.SqlTypes.SqlByte** structure.

**SqlByte**

*Example Syntax:*

Xor

```
[C#]          public          SqlByte(byte          value);
[C++]          public:          SqlByte(unsigned          char          value);
[VB]          Public          Sub          New(ByVal          value          As          Byte)
[JScript]          public          function          SqlByte(value          :          Byte);
```

*Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlByte** structure using the specified byte value. A byte value to be stored in the **System.Data.SqlTypes.SqlByte.Value** property of the new **SqlByte** structure.

IsNull

Xor

```
[C#]          public          bool          IsNull          {get;}
[C++]          public:          __property          bool          get_IsNull();
[VB]          Public          ReadOnly          Property          IsNull          As          Boolean
[JScript]          public          function          get          IsNull()          :          Boolean;
```

*Description*

Indicates whether or not **System.Data.SqlTypes.SqlByte.Value** is null.

Value

Xor

```

1
2 [C#]          public          byte          Value          {get;}
3 [C++]        public:      __property      unsigned      char      get_Value();
4 [VB]        Public      ReadOnly      Property      Value      As      Byte
5 [JScript]    public      function      get      Value()      :      Byte;
6

```

### 7 *Description*

8 Gets the value of the **System.Data.SqlTypes.SqlByte** structure. This  
9 property is read-only The value of the **SqlByte** structure.

### 10 *Add*

```

11
12 [C#]    public    static    SqlByte    Add(SqlByte    x,    SqlByte    y);
13 [C++]    public:    static    SqlByte    Add(SqlByte    x,    SqlByte    y);
14 [VB]    Public Shared Function Add(ByVal x As SqlByte, ByVal y As SqlByte) As
15 SqlByte
16 [JScript] public static function Add(x : SqlByte, y : SqlByte) : SqlByte;
17

```

### 18 *Description*

19 Computes the sum of the two specified **System.Data.SqlTypes.SqlByte**  
20 structures.

21 *Return Value:* A **SqlByte** structure whose **Value** property contains the results of  
22 the addition. A **SqlByte** structure. A **SqlByte** structure.

### 23 *BitwiseAnd*

```

24
25 [C#]    public    static    SqlByte    BitwiseAnd(SqlByte    x,    SqlByte    y);

```

1 [C++] public: static SqlByte BitwiseAnd(SqlByte x, SqlByte y);  
 2 [VB] Public Shared Function BitwiseAnd(ByVal x As SqlByte, ByVal y As  
 3 SqlByte) As SqlByte  
 4 [JScript] public static function BitwiseAnd(x : SqlByte, y : SqlByte) : SqlByte;

#### 6 *Description*

7 Computes the bitwise AND of its **System.Data.SqlTypes.SqlByte**  
 8 operands.

9 *Return Value:* The results of the bitwise AND operation. A **SqlByte** structure. A  
 10 **SqlByte** structure.

#### 11 BitwiseOr

13 [C#] public static SqlByte BitwiseOr(SqlByte x, SqlByte y);  
 14 [C++] public: static SqlByte BitwiseOr(SqlByte x, SqlByte y);  
 15 [VB] Public Shared Function BitwiseOr(ByVal x As SqlByte, ByVal y As  
 16 SqlByte) As SqlByte  
 17 [JScript] public static function BitwiseOr(x : SqlByte, y : SqlByte) : SqlByte;

#### 19 *Description*

20 Computes the bitwise OR of its two **System.Data.SqlTypes.SqlByte**  
 21 operands.

22 *Return Value:* The results of the bitwise OR operation. A **SqlByte** structure. A  
 23 **SqlByte** structure.

#### 24 CompareTo

25

```

1
2 [C#]      public      int      CompareTo(object      value);
3 [C++]     public:     __sealed  int      CompareTo(Object*      value);
4 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
5 Integer
6 [JScript] public  function  CompareTo(value  :  Object)  :  int;
7

```

### *Description*

Compares this instance to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

### *Divide*

```

13
14
15 [C#]  public  static  SqlByte  Divide(SqlByte  x,  SqlByte  y);
16 [C++]  public:  static  SqlByte  Divide(SqlByte  x,  SqlByte  y);
17 [VB] Public Shared Function Divide(ByVal x As SqlByte, ByVal y As SqlByte)
18 As                                     SqlByte
19 [JScript] public static function Divide(x : SqlByte, y : SqlByte) : SqlByte;
20

```

### *Description*

Divides its first **System.Data.SqlTypes.SqlByte** operand by its second.

*Return Value:* A new **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property contains the results of the division. A **SqlByte** structure. A **SqlByte** structure.

## Equals

```
[C#]      public      override      bool      Equals(object      value);  
[C++]      public:      bool      Equals(Object*      value);  
[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean  
[JScript] public override function Equals(value : Object) : Boolean;
```

### *Description*

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlByte.Value** property of the **System.Data.SqlTypes.SqlByte** object.

*Return Value:* **true** if object is an instance of **SqlByte** and the two are equal; otherwise **false**. The object to be compared.

## Equals

```
[C#] public static new SqlBoolean Equals(SqlByte x, SqlByte y);  
[C++] public: static SqlBoolean Equals(SqlByte x, SqlByte y);  
[VB] Shadows Public Shared Function Equals(ByVal x As SqlByte, ByVal y As  
SqlByte) As SqlBoolean  
[JScript] public static hide function Equals(x : SqlByte, y : SqlByte) : SqlBoolean;
```

Performs a logical comparison to determine if a **SqlByte** structure's value is equal to another object.

### *Description*



1 **SqlBoolean;**

2  
3 *Description*

4       Compares two instances of **System.Data.SqlTypes.SqlByte** to determine if  
5 the first is greater than the second.

6 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
7 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the  
8 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
9 instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the  
10 **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlByte**  
11 structure. A **SqlByte** structure.

12       **GreaterThanOrEqual**

13  
14 [C#] public static SqlBoolean GreaterThanOrEqual(SqlByte x, SqlByte y);

15 [C++] public: static SqlBoolean GreaterThanOrEqual(SqlByte x, SqlByte y);

16 [VB] Public Shared Function GreaterThanOrEqual(ByVal x As SqlByte, ByVal y  
17 As SqlByte) As SqlBoolean

18 [JScript] public static function GreaterThanOrEqual(x : SqlByte, y : SqlByte) :

19 **SqlBoolean;**

20  
21 *Description*

22       Compares two **SqlByte** structures to determine if the first is greater than or  
23 equal to the second.

24 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
25 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or

equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**

If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlByte** structure. A **SqlByte** structure.

#### LessThan

```
[C#] public static SqlBoolean LessThan(SqlByte x, SqlByte y);
[C++] public: static SqlBoolean LessThan(SqlByte x, SqlByte y);
[VB] Public Shared Function LessThan(ByVal x As SqlByte, ByVal y As
SqlByte) As SqlBoolean
[JScript] public static function LessThan(x : SqlByte, y : SqlByte) : SqlBoolean;
```

#### Description

Compares two instances of **System.Data.SqlTypes.SqlByte** to determine if the first is less than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlByte** structure. A **SqlByte** structure.

#### LessThanOrEqual

```
[C#] public static SqlBoolean LessThanOrEqual(SqlByte x, SqlByte y);
```

```

1 [C++] public: static SqlBoolean LessThanOrEqual(SqlByte x, SqlByte y);
2 [VB] Public Shared Function LessThanOrEqual(ByVal x As SqlByte, ByVal y As
3 SqlByte) As SqlBoolean
4 [JScript] public static function LessThanOrEqual(x : SqlByte, y : SqlByte) :
5 SqlBoolean;

```

#### 7 *Description*

8 Compares two instances of **System.Data.SqlTypes.SqlByte** to determine if  
9 the first is less than or equal to the second. A **SqlByte** structure. A **SqlByte**  
10 structure.

#### 11 *Mod*

```

12
13 [C#] public static SqlByte Mod(SqlByte x, SqlByte y);
14 [C++] public: static SqlByte Mod(SqlByte x, SqlByte y);
15 [VB] Public Shared Function Mod(ByVal x As SqlByte, ByVal y As SqlByte) As
16 SqlByte
17 [JScript] public static function Mod(x : SqlByte, y : SqlByte) : SqlByte;

```

#### 19 *Description*

20 Computes the remainder after dividing its first  
21 **System.Data.SqlTypes.SqlByte** operand by its second.

22 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
23 contains the remainder. A **SqlByte** structure. A **SqlByte** structure.

#### 24 *Multiply*

25

```

1
2 [C#] public static SqlByte Multiply(SqlByte x, SqlByte y);
3 [C++] public: static SqlByte Multiply(SqlByte x, SqlByte y);
4 [VB] Public Shared Function Multiply(ByVal x As SqlByte, ByVal y As SqlByte)
5 As SqlByte
6 [JScript] public static function Multiply(x : SqlByte, y : SqlByte) : SqlByte;
7

```

### *Description*

Computes the product of the two **System.Data.SqlTypes.SqlByte** operands.

**Return Value:** A new **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property contains the product of the multiplication. A **SqlByte** structure. A **SqlByte** structure.

### **NotEquals**

```

14
15
16 [C#] public static SqlBoolean NotEquals(SqlByte x, SqlByte y);
17 [C++] public: static SqlBoolean NotEquals(SqlByte x, SqlByte y);
18 [VB] Public Shared Function NotEquals(ByVal x As SqlByte, ByVal y As
19 SqlByte) As SqlBoolean
20 [JScript] public static function NotEquals(x : SqlByte, y : SqlByte) : SqlBoolean;
21

```

### *Description*

Compares two instances of **System.Data.SqlTypes.SqlByte** for equality.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or

**System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlByte** structure. A **SqlByte** structure.

#### OnesComplement

```
[C#]      public      static      SqlByte      OnesComplement(SqlByte      x);
[C++]     public:     static      SqlByte      OnesComplement(SqlByte      x);
[VB]      Public Shared Function OnesComplement(ByVal x As SqlByte) As SqlByte
[JScript] public static function OnesComplement(x : SqlByte) : SqlByte;
```

#### Description

The ones complement operator performs a bitwise one's complement operation on its **System.Data.SqlTypes.SqlByte** operand.

**Return Value:** A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property contains the ones complement of the **SqlByte** parameter. A **SqlByte** structure.

#### op\_Addition

```
[C#]      public      static      SqlByte      operator      +(SqlByte      x,      SqlByte      y);
[C++]     public:     static      SqlByte      op_Addition(SqlByte      x,      SqlByte      y);
[VB]      returnValue      =      SqlByte.op_Addition(x,      y)
[JScript]      returnValue      =      x      +      y;
```

#### Description

Computes the sum of the two specified **System.Data.SqlTypes.SqlByte** structures.

*Return Value:* A **SqlByte** whose **System.Data.SqlTypes.SqlByte.Value** property contains the sum of the two operands. A **SqlByte** structure. A **SqlByte** structure.

op\_BitwiseAnd

[C#] public static SqlByte operator &(SqlByte x, SqlByte y);

[C++] public: static SqlByte op\_BitwiseAnd(SqlByte x, SqlByte y);

[VB] returnValue = SqlByte.op\_BitwiseAnd(x, y)

[JScript] returnValue = x & y;

#### *Description*

Computes the bitwise AND of its **System.Data.SqlTypes.SqlByte** operands.

*Return Value:* The results of the bitwise AND operation. A **SqlByte** structure. A **SqlByte** structure.

op\_BitwiseOr

[C#] public static SqlByte operator |(SqlByte x, SqlByte y);

[C++] public: static SqlByte op\_BitwiseOr(SqlByte x, SqlByte y);

[VB] returnValue = SqlByte.op\_BitwiseOr(x, y)

[JScript] returnValue = x | y;

#### *Description*

Computes the bitwise OR of its two **System.Data.SqlTypes.SqlByte** operands.

*Return Value:* The results of the bitwise OR operation. A **SqlByte** structure. A **SqlByte** structure.

op\_Division

[C#] public static SqlByte operator /(SqlByte x, SqlByte y);

[C++] public: static SqlByte op\_Division(SqlByte x, SqlByte y);

[VB] returnValue = SqlByte.op\_Division(x, y)

[JScript] returnValue = x / y;

#### *Description*

Divides its first **System.Data.SqlTypes.SqlByte** operand by its second.

*Return Value:* A new **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property contains the results of the division. A **SqlByte** structure. A **SqlByte** structure.

op\_Equality

[C#] public static SqlBoolean operator ==(SqlByte x, SqlByte y);

[C++] public: static SqlBoolean op\_Equality(SqlByte x, SqlByte y);

[VB] returnValue = SqlByte.op\_Equality(x, y)

[JScript] returnValue = x == y;

#### *Description*

Performs a logical comparison of two **System.Data.SqlTypes.SqlByte** structures to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlByte** structure. A **SqlByte** structure.

op\_ExclusiveOr

```
[C#] public static SqlByte operator ^(SqlByte x, SqlByte y);
[C++] public: static SqlByte op_ExclusiveOr(SqlByte x, SqlByte y);
[VB] returnValue = SqlByte.op_ExclusiveOr(x, y)
[JScript] returnValue = x ^ y;
```

### Description

Performs a bitwise exclusive-OR operation on the supplied parameters.

*Return Value:* The results of the bitwise XOR operation. A **SqlByte** structure. A **SqlByte** structure.

op\_Explicit

```
[C#] public static explicit operator SqlByte(SqlBoolean x);
[C++] public: static SqlByte op_Explicit(SqlBoolean x);
[VB] returnValue = SqlByte.op_Explicit(x)
[JScript] returnValue = SqlByte(x);
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Converts the **System.Data.SqlTypes.SqlBoolean** parameter to a **System.Data.SqlTypes.SqlByte**.

*Return Value:* A **SqlByte** whose **System.Data.SqlTypes.SqlByte.Value** property equals the **System.Data.SqlTypes.SqlBit.ByteValue** of the supplied **SqlBoolean** parameter. The **SqlBoolean** parameter to be converted to a **SqlByte**.

```

op_Explicit

[C#]      public      static      explicit      operator      byte(SqlByte      x);
[C++]      public:      static      unsigned      char      op_Explicit();
[VB]      returnValue      =      SqlByte.op_Explicit(x)
[JScript]      returnValue      =      Byte(x);

```

*Description*

Converts the supplied **System.Data.SqlTypes.SqlByte** structure to a byte.

*Return Value:* A byte whose value equals the **System.Data.SqlTypes.SqlByte.Value** property of the **SqlByte** parameter. The **SqlByte** structure to be converted to a byte.

```

op_Explicit

[C#]      public      static      explicit      operator      SqlByte(SqlDecimal      x);
[C++]      public:      static      SqlByte      op_Explicit(SqlDecimal      x);
[VB]      returnValue      =      SqlByte.op_Explicit(x)
[JScript]      returnValue      =      SqlByte(x);

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Converts the supplied **System.Data.SqlTypes.SqlDecimal** to **System.Data.SqlTypes.SqlByte**.

*Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property is equal to the **System.Data.SqlTypes.SqlDecimal.Value** of the **Decimal** parameter. A **SqlDecimal** structure.

```

op_Explicit

[C#]    public static explicit operator SqlByte(SqlDouble x);
[C++]   public: static SqlByte op_Explicit(SqlDouble x);
[VB]    returnValue = SqlByte.op_Explicit(x)
[JScript]    returnValue = SqlByte(x);

```

*Description*

Converts the supplied **System.Data.SqlTypes.SqlDouble** to **System.Data.SqlTypes.SqlByte**.

*Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property is equal to the **System.Data.SqlTypes.SqlDouble.Value** of the **Double** parameter. A **SqlDouble** structure.

```

op_Explicit

[C#]    public static explicit operator SqlByte(SqlInt16 x);
[C++]   public: static SqlByte op_Explicit(SqlInt16 x);
[VB]    returnValue = SqlByte.op_Explicit(x)

```

1 [JScript]                      returnValue                      =                      SqlByte(x);

2

3 *Description*

4            Converts    the    **System.Data.SqlTypes.SqlInt16**    parameter    to    a  
5 **System.Data.SqlTypes.SqlByte**

6 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
7 property is equal to the **System.Data.SqlTypes.SqlInt16.Value** of the **SqlInt16**  
8 parameter. A **SqlInt16** structure.

9            op\_Explicit

10

11 [C#]        public        static        explicit        operator        SqlByte(SqlInt32    x);

12 [C++]        public:        static        SqlByte        op\_Explicit(SqlInt32    x);

13 [VB]                      returnValue                      =                      SqlByte.op\_Explicit(x)

14 [JScript]                      returnValue                      =                      SqlByte(x);

15

16 *Description*

17            Converts    the    supplied    **System.Data.SqlTypes.SqlInt32**    to  
18 **System.Data.SqlTypes.SqlByte**

19 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
20 property is equal to the **System.Data.SqlTypes.SqlInt32.Value** of the **SqlInt32**  
21 parameter. A **SqlInt32** structure.

22            op\_Explicit

23

24 [C#]        public        static        explicit        operator        SqlByte(SqlInt64    x);

25 [C++]        public:        static        SqlByte        op\_Explicit(SqlInt64    x);

1 [VB]                   returnValue                   =                   SqlByte.op\_Explicit(x)  
 2 [JScript]                   returnValue                   =                   SqlByte(x);  
 3

4 *Description*

5       Converts       the       supplied       **System.Data.SqlTypes.SqlInt64**       to  
 6 **System.Data.SqlTypes.SqlByte**

7 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
 8 property is equal to the **System.Data.SqlTypes.SqlInt64.Value** of the **SqlInt64**  
 9 parameter. A **SqlInt64** structure.

10       op\_Explicit

11  
 12 [C#]       public       static       explicit       operator       SqlByte(SqlMoney       x);  
 13 [C++]       public:       static       SqlByte       op\_Explicit(SqlMoney       x);  
 14 [VB]                   returnValue                   =                   SqlByte.op\_Explicit(x)  
 15 [JScript]                   returnValue                   =                   SqlByte(x);  
 16

17 *Description*

18       Converts       the       **System.Data.SqlTypes.SqlMoney**       parameter       to       a  
 19 **System.Data.SqlTypes.SqlByte**

20 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
 21 property is equal to the **System.Data.SqlTypes.SqlMoney.Value** of the  
 22 **SqlMoney** parameter. A **SqlMoney** structure.

23       op\_Explicit

24  
 25 [C#]       public       static       explicit       operator       SqlByte(SqlSingle       x);

```

1  [C++]      public:      static      SqlByte      op_Explicit(SqlSingle      x);
2  [VB]              returnValue      =              SqlByte.op_Explicit(x)
3  [JScript]              returnValue      =              SqlByte(x);

```

#### 5 *Description*

6 Converts the supplied **System.Data.SqlTypes.SqlSingle** structure to  
7 **System.Data.SqlTypes.SqlByte**

8 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
9 property is equal to the **System.Data.SqlTypes.SqlSingle.Value** of the **SqlSingle**  
10 parameter. A **SqlSingle** structure.

11 op\_Explicit

```

12
13 [C#]      public      static      explicit      operator      SqlByte(SqlString      x);
14 [C++]      public:      static      SqlByte      op_Explicit(SqlString      x);
15 [VB]              returnValue      =              SqlByte.op_Explicit(x)
16 [JScript]              returnValue      =              SqlByte(x);

```

#### 18 *Description*

19 Converts the supplied **System.Data.SqlTypes.SqlString** to  
20 **System.Data.SqlTypes.SqlByte**

21 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
22 property is equal to the numeric value represented by the **SqlString** . An instance  
23 of the **SqlString** class.

24 op\_GreaterThan

25

```

[C#] public static SqlBoolean operator >(SqlByte x, SqlByte y);
[C++] public: static SqlBoolean op_GreaterThan(SqlByte x, SqlByte y);
[VB]     returnValue      =      SqlByte.op_GreaterThan(x,      y)
[JScript]     returnValue      =      x      >      y;
    
```

### Description

Compares two instances of **System.Data.SqlTypes.SqlByte** to determine if the first is greater than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlByte** structure. A **SqlByte** structure.

op\_GreaterThanOrEqual

```

[C#] public static SqlBoolean operator >=(SqlByte x, SqlByte y);
[C++] public: static SqlBoolean op_GreaterThanOrEqual(SqlByte x, SqlByte y);
[VB]     returnValue      =      SqlByte.op_GreaterThanOrEqual(x,      y)
[JScript]     returnValue      =      x      >=      y;
    
```

### Description

Compares two instances of **System.Data.SqlTypes.SqlByte** to determine if the first is greater than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**.

If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlByte** structure. A **System.Data.SqlTypes.SqlByte** structure.

#### op\_Implicit

```
[C#]      public static implicit operator SqlByte(byte x);
[C++]     public: static SqlByte op_Implicit(unsigned char x);
[VB]      returnValue = SqlByte.op_Implicit(x)
[JScript]      returnValue = x;
```

#### Description

Converts the supplied byte value to a **System.Data.SqlTypes.SqlByte**.

*Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property is equal to the supplied parameter. A byte value to be converted to **SqlByte**.

#### op\_Inequality

```
[C#]      public static SqlBoolean operator !=(SqlByte x, SqlByte y);
[C++]     public: static SqlBoolean op_Inequality(SqlByte x, SqlByte y);
[VB]      returnValue = SqlByte.op_Inequality(x, y)
[JScript]      returnValue = x != y;
```

## Description

Compares two instances of **System.Data.SqlTypes.SqlByte** for equality.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlByte** structure. A **SqlByte** structure.

op\_LessThan

[C#]            public            static            SqlBoolean            operator  
[C++]   public:   static   SqlBoolean   op\_LessThan(SqlByte x,   SqlByte y);  
[VB]            returnValue            =            SqlByte.op\_LessThan(x,            y)  
[JScript]            returnValue            =            x            <            y;

## Description

Compares two instances of **System.Data.SqlTypes.SqlByte** to determine if the first is less than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlByte** structure. A **SqlByte** structure.

## op\_LessThanOrEqual

```
[C#] public static SqlBoolean operator <=(SqlByte x, SqlByte y);
[C++] public: static SqlBoolean op_LessThanOrEqual(SqlByte x, SqlByte y);
[VB]     returnValue      =      SqlByte.op_LessThanOrEqual(x,      y)
[JScript]     returnValue      =      x      <=      y;
```

### *Description*

Compares two instances of **System.Data.SqlTypes.SqlByte** to determine if the first is less than or equal to the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is: **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **SqlByte** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlByte** structure. A **SqlByte** structure.

## op\_Modulus

```
[C#] public static SqlByte operator %(SqlByte x, SqlByte y);
[C++] public: static SqlByte op_Modulus(SqlByte x, SqlByte y);
[VB]     returnValue      =      SqlByte.op_Modulus(x,      y)
[JScript]     returnValue      =      x      %      y;
```

### *Description*

1 Computes the remainder after dividing its first  
 2 **System.Data.SqlTypes.SqlByte** operand by its second.

3 *Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value**  
 4 contains the remainder. A **SqlByte** structure. A **SqlByte** structure.

5 op\_Multiply

6  
 7 [C#] public static SqlByte operator \*(SqlByte x, SqlByte y);

8 [C++] public: static SqlByte op\_Multiply(SqlByte x, SqlByte y);

9 [VB] returnValue = SqlByte.op\_Multiply(x, y)

10 [JScript] returnValue = x \* y;

11  
 12 *Description*

13 Computes the product of the two **System.Data.SqlTypes.SqlByte**  
 14 operands.

15 *Return Value:* A new **SqlByte** structure whose  
 16 **System.Data.SqlTypes.SqlByte.Value** property contains the product of the  
 17 multiplication. A **SqlByte** structure. A **SqlByte** structure.

18 op\_OnesComplement

19  
 20 [C#] public static SqlByte operator ~(SqlByte x);

21 [C++] public: static SqlByte op\_OnesComplement(SqlByte x);

22 [VB] returnValue = SqlByte.op\_OnesComplement(x)

23 [JScript] returnValue = ~x;

24  
 25 *Description*

The ones complement operator performs a bitwise one's complement operation on its **System.Data.SqlTypes.SqlByte** operand.

*Return Value:* A **SqlByte** structure whose **System.Data.SqlTypes.SqlByte.Value** property contains the ones complement of the **SqlByte** parameter. A **SqlByte** structure.

**op\_Subtraction**

[C#]    public    static    SqlByte    operator    -(SqlByte    x,    SqlByte    y);

[C++]   public:   static    SqlByte    op\_Subtraction(SqlByte    x,    SqlByte    y);

[VB]        returnValue        =        SqlByte.op\_Subtraction(x,        y)

[JScript]        returnValue        =        x        -        y;

### *Description*

Subtracts the second **System.Data.SqlTypes.SqlByte** operand from the first.

*Return Value:* The results of subtracting the second **SqlByte** operand from the first. A **SqlByte** structure. A **SqlByte** structure.

**Parse**

[C#]        public        static        SqlByte        Parse(string        s);

[C++]        public:        static        SqlByte        Parse(String\*        s);

[VB]    Public    Shared    Function    Parse(ByVal    s    As    String)    As    SqlByte

[JScript]    public    static    function    Parse(s    :    String)    :    SqlByte;

### *Description*

```

1      [ .]
2      Subtract
3
4      [C#]    public    static    SqlByte    Subtract(SqlByte    x,    SqlByte    y);
5      [C++]   public:   static    SqlByte    Subtract(SqlByte    x,    SqlByte    y);
6      [VB]    Public Shared Function Subtract(ByVal x As SqlByte, ByVal y As SqlByte)
7      As                                             SqlByte
8      [JScript] public static function Subtract(x : SqlByte, y : SqlByte) : SqlByte;
9

```

#### 10 *Description*

11 Subtracts the second **System.Data.SqlTypes.SqlByte** operand from the  
12 first.

13 *Return Value:* The results of subtracting the second **SqlByte** operand from the  
14 first. A **SqlByte** structure. A **SqlByte** structure.

#### 15 **ToSqlBoolean**

```

16
17      [C#]          public          SqlBoolean          ToSqlBoolean();
18      [C++]         public:         SqlBoolean          ToSqlBoolean();
19      [VB]          Public          Function            ToSqlBoolean()    As          SqlBoolean
20      [JScript]     public          function            ToSqlBoolean()    :          SqlBoolean;
21

```

#### 22 *Description*

23 Converts this **System.Data.SqlTypes.SqlByte** structure to  
24 **System.Data.SqlTypes.SqlBoolean**.

25 *Return Value:* A **SqlBoolean** that will be

**System.Data.SqlTypes.SqlBoolean.True** if the **System.Data.SqlTypes.Value** of the **SqlByte** structure is non-zero, **False** if the **SqlByte** is zero and **Null** if the **SqlByte** is **Null**.

**ToSqlDecimal**

```
[C#]          public          SqlDecimal          ToSqlDecimal();
[C++]          public:          SqlDecimal          ToSqlDecimal();
[VB]    Public    Function    ToSqlDecimal()    As    SqlDecimal
[JScript]    public    function    ToSqlDecimal()    :    SqlDecimal;
```

*Description*

Converts this **System.Data.SqlTypes.SqlByte** structure to **System.Data.SqlTypes.SqlDecimal**.

**Return Value:** A **SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** equals the **System.Data.SqlTypes.SqlByte.Value** of this **SqlByte** structure.

**ToSqlDouble**

```
[C#]          public          SqlDouble          ToSqlDouble();
[C++]          public:          SqlDouble          ToSqlDouble();
[VB]    Public    Function    ToSqlDouble()    As    SqlDouble
[JScript]    public    function    ToSqlDouble()    :    SqlDouble;
```

*Description*

1        Converts        this        **System.Data.SqlTypes.SqlByte**        structure        to  
2        **System.Data.SqlTypes.SqlDouble**        .

3        *Return Value:* A **SqlDouble** structure with the same value as this **SqlByte** .

4        **ToSqlInt16**

6        [C#]                public                **SqlInt16**                **ToSqlInt16();**

7        [C++]                public:                **SqlInt16**                **ToSqlInt16();**

8        [VB]        Public        Function        **ToSqlInt16()**        As        **SqlInt16**

9        [JScript]        public        function        **ToSqlInt16()**        :        **SqlInt16;**

11        *Description*

12        Converts this **SqlByte** structure to **System.Data.SqlTypes.SqlInt16** .

13        *Return Value:* A **SqlInt16** structure with the same value as this **SqlByte** .

14        **ToSqlInt32**

16        [C#]                public                **SqlInt32**                **ToSqlInt32();**

17        [C++]                public:                **SqlInt32**                **ToSqlInt32();**

18        [VB]        Public        Function        **ToSqlInt32()**        As        **SqlInt32**

19        [JScript]        public        function        **ToSqlInt32()**        :        **SqlInt32;**

21        *Description*

22        Converts        this        **System.Data.SqlTypes.SqlByte**        to  
23        **System.Data.SqlTypes.SqlInt32**        .

24        *Return Value:* A **SqlInt32** structure with the same value as this **SqlByte** .

25        **ToSqlInt64**

```

1
2 [C#]          public          SqlInt64          ToSqlInt64();
3 [C++]         public:         SqlInt64          ToSqlInt64();
4 [VB]         Public          Function          ToSqlInt64()    As          SqlInt64
5 [JScript]     public          function          ToSqlInt64()    :          SqlInt64;

```

#### 6 7 *Description*

8        Converts        this        **System.Data.SqlTypes.SqlByte**        structure        to  
9        **System.Data.SqlTypes.SqlInt64**  
10        *Return Value:* A **SqlInt64** structure who **System.Data.SqlTypes.SqlInt64.Value**  
11        equals the **System.Data.SqlTypes.SqlByte.Value** of this **SqlByte** .

12        ToSqlMoney

```

13
14 [C#]          public          SqlMoney          ToSqlMoney();
15 [C++]         public:         SqlMoney          ToSqlMoney();
16 [VB]         Public          Function          ToSqlMoney()    As          SqlMoney
17 [JScript]     public          function          ToSqlMoney()    :          SqlMoney;

```

#### 18 19 *Description*

20        Converts        this        **System.Data.SqlTypes.SqlByte**        structure        to  
21        **System.Data.SqlTypes.SqlMoney**  
22        *Return Value:* A        **SqlMoney**        structure        whose  
23        **System.Data.SqlTypes.SqlMoney.Value**        equals        the  
24        **System.Data.SqlTypes.SqlByte.Value** of this **SqlByte** structure.

25        ToSqlSingle

2	[C#]	public	SqlSingle	ToSqlSingle();
3	[C++]	public:	SqlSingle	ToSqlSingle();
4	[VB]	Public	Function	ToSqlSingle() As SqlSingle
5	[JScript]	public	function	ToSqlSingle() : SqlSingle;

### Description

Converts this **System.Data.SqlTypes.SqlByte** structure to **System.Data.SqlTypes.SqlSingle**.

**Return Value:** A **SqlSingle** structure that has the same **System.Data.SqlTypes.SqlSingle.Value** as this **SqlByte** structure.

## ToSqlString

14	[C#]	public	SqlString	ToSqlString();
15	[C++]	public:	SqlString	ToSqlString();
16	[VB]	Public	Function	ToSqlString() As SqlString
17	[JScript]	public	function	ToSqlString() : SqlString;

### Description

Converts this instance of **System.Data.SqlTypes.SqlByte** to **System.Data.SqlTypes.SqlString**.

**Return Value:** A **SqlString** containing the string representation of the **SqlByte** structure's **System.Data.SqlTypes.SqlByte.Value** . .

## ToString

```

1
2 [C#]          public          override          string          ToString();
3 [C++]          public:          String*          ToString();
4 [VB]  Overrides  Public  Function  ToString()  As  String
5 [JScript] public  override  function  ToString()  :  String;  Converts  a
6 System.Data.SqlTypes.SqlByte          to          a          string.
7

```

### *Description*

Converts this **System.Data.SqlTypes.SqlByte** structure to a **System.String**

*Return Value:* A string containing the **System.Data.SqlTypes.SqlByte.Value** of the **SqlByte** . If the **Value** is null, the **String** will be a null string.

### **Xor**

```

15 [C#]  public  static  SqlByte  Xor(SqlByte  x,  SqlByte  y);
16 [C++]  public:  static  SqlByte  Xor(SqlByte  x,  SqlByte  y);
17 [VB] Public Shared Function Xor(ByVal x As SqlByte, ByVal y As SqlByte) As
18 SqlByte
19 [JScript] public static function Xor(x : SqlByte, y : SqlByte) : SqlByte;
20

```

### *Description*

Performs a bitwise exclusive-OR operation on the supplied parameters.

*Return Value:* The results of the XOR operation. A **SqlByte** structure. A **SqlByte** structure.

**SqlCompareOptions** enumeration (System.Data.SqlTypes)

1 Xor

2

3

4 *Description*

5 Specifies the compare option values for a

6 **System.Data.SqlTypes.SqlString** structure.

7 Xor

8

9 [C#] public const SqlCompareOptions BinarySort;

10 [C++] public: const SqlCompareOptions BinarySort;

11 [VB] Public Const BinarySort As SqlCompareOptions

12 [JScript] public var BinarySort : SqlCompareOptions;

13

14 *Description*

15 Specifies that sorts should be based on a characters numeric value rather

16 than its alphabetic value.

17 Xor

18

19 [C#] public const SqlCompareOptions IgnoreCase;

20 [C++] public: const SqlCompareOptions IgnoreCase;

21 [VB] Public Const IgnoreCase As SqlCompareOptions

22 [JScript] public var IgnoreCase : SqlCompareOptions;

23

24 *Description*

25 Specifies that SqlString comparisons must ignore case.

## Xor

```
[C#]      public      const      SqlCompareOptions      IgnoreKanaType;
[C++]     public:     const      SqlCompareOptions      IgnoreKanaType;
[VB]      Public      Const      IgnoreKanaType      As      SqlCompareOptions
[JScript] public      var      IgnoreKanaType      :      SqlCompareOptions;
```

### *Description*

Specifies that the string comparison must ignore the Kana type. Kana type refers to Japanese hiragana and katakana characters, which represent phonetic sounds in the Japanese language. Hiragana is used for native Japanese expressions and words, while katakana is used for words borrowed from other languages, such as "computer" or "internet". A phonetic sound can be expressed in both hiragana and katakana. If this value is selected, the hiragana character for one sound is considered equal to the katakana character for the same sound.

## Xor

```
[C#]      public      const      SqlCompareOptions      IgnoreNonSpace;
[C++]     public:     const      SqlCompareOptions      IgnoreNonSpace;
[VB]      Public      Const      IgnoreNonSpace      As      SqlCompareOptions
[JScript] public      var      IgnoreNonSpace      :      SqlCompareOptions;
```

### *Description*

Specifies that the string comparison must ignore nonspace combining characters, such as diacritics. The Unicode Standard defines combining characters

as characters that are combined with base characters to produce a new character. Non-space combining characters do not take up character space by themselves when rendered. For more information on non-space combining characters, see the Unicode Standard at <http://www.unicode.org>.

#### Xor

[C#]	public	const	SqlCompareOptions	IgnoreWidth;
[C++]	public:	const	SqlCompareOptions	IgnoreWidth;
[VB]	Public	Const	IgnoreWidth	As SqlCompareOptions
[JScript]	public	var	IgnoreWidth	: SqlCompareOptions;

#### Description

Specifies that the string comparison must ignore the character width. For example, Japanese katakana characters can be written as full-width or half-width and, if this value is selected, the katakana characters written as full-width are considered equal to the same characters written in half-width.

#### Xor

[C#]	public	const	SqlCompareOptions	None;
[C++]	public:	const	SqlCompareOptions	None;
[VB]	Public	Const	None	As SqlCompareOptions
[JScript]	public	var	None	: SqlCompareOptions;

#### Description

Specifies the default option settings for **SqlString** comparisons.

SQL Server 2000

1	SqlDateTime structure (System.Data.SqlTypes)						
2	ToString						
3							
4							
5	<i>Description</i>						
6	Represents the date and time data ranging in value from January 1, 1753 to						
7	December 31, 9999 to an accuracy of 3.33 milliseconds to be stored in or retrieved						
8	from a database.						
9	ToString						
10							
11	[C#]	public	static	readonly	SqlDateTime	MaxValue;	
12	[C++]	public:	static		SqlDateTime	MaxValue;	
13	[VB]	Public	Shared	ReadOnly	MaxValue	As	SqlDateTime
14	[JScript]	public	static	var	MaxValue	:	SqlDateTime;
15							
16	<i>Description</i>						
17	Represents the maximum valid date value for a						
18	<b>System.Data.SqlTypes.SqlDateTime</b> structure.						
19	The maximum valid date for a <b>SqlDateTime</b> structure is December 31,						
20	9999.						
21	ToString						
22							
23	[C#]	public	static	readonly	SqlDateTime	MinValue;	
24	[C++]	public:	static		SqlDateTime	MinValue;	
25	[VB]	Public	Shared	ReadOnly	MinValue	As	SqlDateTime

[JScript] public static var MinValue : SqlDateTime;

*Description*

Represents the minimum valid date value for a **System.Data.SqlTypes.SqlDateTime** structure.

The minimum valid date for a **SqlDateTime** structure is January 1, 1753.

**ToString**

[C#] public static readonly SqlDateTime Null;

[C++] public: static SqlDateTime Null;

[VB] Public Shared ReadOnly Null As SqlDateTime

[JScript] public static var Null : SqlDateTime;

*Description*

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlDateTime.Value** property of an instance of the **System.Data.SqlTypes.SqlDateTime** structure.

**Null** functions as a constant for the **SqlDateTime** structure.

**ToString**

[C#] public static readonly int SQLTicksPerHour;

[C++] public: static int SQLTicksPerHour;

[VB] Public Shared ReadOnly SQLTicksPerHour As Integer

[JScript] public static var SQLTicksPerHour : int;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

A constant whose value is the number of ticks equivalent to one hour.

ToString

```
[C#]      public      static      readonly      int      SQLTicksPerMinute;
[C++]      public:      static      int      SQLTicksPerMinute;
[VB]      Public      Shared      ReadOnly      SQLTicksPerMinute      As      Integer
[JScript]      public      static      var      SQLTicksPerMinute      :      int;
```

*Description*

A constant whose value is the number of ticks equivalent to one minute.

ToString

```
[C#]      public      static      readonly      int      SQLTicksPerSecond;
[C++]      public:      static      int      SQLTicksPerSecond;
[VB]      Public      Shared      ReadOnly      SQLTicksPerSecond      As      Integer
[JScript]      public      static      var      SQLTicksPerSecond      :      int;
```

*Description*

A constant whose value is the number of ticks equivalent to one second.

SqlDateTime

*Example Syntax:*

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]          public          SqlDateTime(DateTime          value);
[C++]          public:          SqlDateTime(DateTime          value);
[VB]    Public    Sub    New(ByVal    value    As    DateTime)
[JavaScript] public function SqlDateTime(value : DateTime); Initializes a new
instance    of    the    System.Data.SqlTypes.SqlDateTime    structure.
```

*Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlDateTime** structure using the specified **System.DateTime** value. A **System.DateTime** structure.

SqlDateTime

*Example Syntax:*

ToString

```
[C#]    public    SqlDateTime(int    dayTicks,    int    timeTicks);
[C++]    public:    SqlDateTime(int    dayTicks,    int    timeTicks);
[VB] Public Sub New(ByVal dayTicks As Integer, ByVal timeTicks As Integer)
[JavaScript] public function SqlDateTime(dayTicks : int, timeTicks : int);
```

*Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlDateTime** structure using the supplied parameters. An integer value that represents the date as ticks. An integer value that represents the time as ticks.

SqlDateTime

*Example Syntax:*

ToString

```
[C#]    public    SqlDateTime(int    year,    int    month,    int    day);  
[C++]  public:    SqlDateTime(int    year,    int    month,    int    day);  
[VB]   Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal  
day                                         As                                         Integer)  
[JScript] public function SqlDateTime(year : int, month : int, day : int);
```

*Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlDateTime** structure using the supplied parameters to initialize the year, month, day. An integer representing the year of the of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the month of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the day number of the new **System.Data.SqlTypes.SqlDateTime** structure.

SqlDateTime

*Example Syntax:*

ToString

```
[C#] public SqlDateTime(int year, int month, int day, int hour, int minute, int  
second);  
[C++] public: SqlDateTime(int year, int month, int day, int hour, int minute, int  
second);
```

```

[VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal
day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second
As Integer)
[JScript] public function SqlDateTime(year : int, month : int, day : int, hour : int,
minute : int, second : int);
    
```

### *Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlDateTime** structure using the supplied parameters to initialize the year, month, day, hour, minute, and second of the new structure. An integer value representing the year of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the month of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the day of the month of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the hour of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the minute of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the second of the new **System.Data.SqlTypes.SqlDateTime** structure.

**SqlDateTime**

*Example Syntax:*

**ToString**

```

[C#] public SqlDateTime(int year, int month, int day, int hour, int minute, int
second, double millisecond);
[C++] public: SqlDateTime(int year, int month, int day, int hour, int minute, int
    
```

```

1 second, double millisecond);
2 [VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal
3 day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second
4 As Integer, ByVal millisecond As Double)
5 [JScript] public function SqlDateTime(year : int, month : int, day : int, hour : int,
6 minute : int, second : int, millisecond : double);
7

```

### *Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlDateTime** structure using the supplied parameters to initialize the year, month, day, hour, minute, second, and millisecond of the new structure. An integer value representing the year of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the month of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the day of the month of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the hour of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the minute of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the second of the new **System.Data.SqlTypes.SqlDateTime** structure. An double value representing the millisecond of the new **System.Data.SqlTypes.SqlDateTime** structure.

SqlDateTime

*Example Syntax:*

ToString

```

1  [C#] public SqlDateTime(int year, int month, int day, int hour, int minute, int
2  second,                                int                                bilisecond);
3
4  [C++] public: SqlDateTime(int year, int month, int day, int hour, int minute, int
5  second,                                int                                bilisecond);
6
7  [VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal
8  day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second
9  As Integer, ByVal bilisecond As Integer)
10 [JScript] public function SqlDateTime(year : int, month : int, day : int, hour : int,
11 minute : int, second : int, bilisecond : int);
12

```

### *Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlDateTime** structure using the supplied parameters to initialize the year, month, day, hour, minute, second, and bilisecond of the new structure. An integer value representing the year of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the month of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the day of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the hour of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the minute of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the second of the new **System.Data.SqlTypes.SqlDateTime** structure. An integer value representing the bilisecond (billionth of a second) of the new **System.Data.SqlTypes.SqlDateTime** structure.

1	DayTicks								
2	ToString								
3									
4	[C#]	public	int	DayTicks	{get;}				
5	[C++]	public:	__property	int	get_DayTicks();				
6	[VB]	Public	ReadOnly	Property	DayTicks	As	Integer		
7	[JScript]	public	function	get	DayTicks()	:	int;		
8									
9	<i>Description</i>								
10	Gets the number of ticks representing the date of this								
11	<b>System.Data.SqlTypes.SqlDateTime</b> structure.								
12	IsNull								
13	ToString								
14									
15	[C#]	public	bool	IsNull	{get;}				
16	[C++]	public:	__property	bool	get_IsNull();				
17	[VB]	Public	ReadOnly	Property	IsNull	As	Boolean		
18	[JScript]	public	function	get	IsNull()	:	Boolean;		
19									
20	<i>Description</i>								
21	Gets a value indicating whether the Value property of the SqlDateTime								
22	structure is null.								
23	TimeTicks								
24	ToString								
25									

```

1
2 [C#]          public          int          TimeTicks          {get;}
3 [C++]          public:          __property          int          get_TimeTicks();
4 [VB]    Public    ReadOnly    Property    TimeTicks    As    Integer
5 [JScript]    public    function    get    TimeTicks()    :    int;
6

```

*Description*

Gets the number of ticks representing the time of this **System.Data.SqlTypes.SqlDateTime** structure.

Value  
ToString

```

13 [C#]          public          DateTime          Value          {get;}
14 [C++]          public:          __property          DateTime          get_Value();
15 [VB]    Public    ReadOnly    Property    Value    As    DateTime
16 [JScript]    public    function    get    Value()    :    DateTime;
17

```

*Description*

Gets the value of the **System.Data.SqlTypes.SqlDateTime** structure. This property is read-only.

CompareTo

```

23 [C#]          public          int          CompareTo(object          value);
24 [C++]          public:          __sealed          int          CompareTo(Object*          value);
25 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

```

Integer

[JScript] public function CompareTo(value : Object) : int;

*Description*

Compares this instance to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object\* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean;

*Description*

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlDateTime.Value** property of the **System.Data.SqlTypes.SqlDateTime** object.

*Return Value:* **true** if object is an instance of **System.Data.SqlTypes.SqlDateTime** and the two are equal; otherwise **false** . The object to be compared.

Equals

[C#] public static new SqlBoolean Equals(SqlDateTime x, SqlDateTime y);

```

1 [C++] public: static SqlBoolean Equals(SqlDateTime x, SqlDateTime y);
2 [VB] Shadows Public Shared Function Equals(ByVal x As SqlDateTime, ByVal y
3 As SqlDateTime) As SqlBoolean
4 [JScript] public static hide function Equals(x : SqlDateTime, y : SqlDateTime) :
5 SqlBoolean;

```

#### 7 *Description*

8 Performs a logical comparison of two  
9 **System.Data.SqlTypes.SqlDateTime** structures to determine if they are equal.

#### 10 **GetHashCode**

```

11
12 [C#] public override int GetHashCode();
13 [C++] public: int GetHashCode();
14 [VB] Overrides Public Function GetHashCode() As Integer
15 [JScript] public override function GetHashCode() : int;

```

#### 17 *Description*

18 Gets the hash code for this instance.

19 *Return Value:* A 32-bit signed integer hash code.

#### 20 **GreaterThan**

```

21
22 [C#] public static SqlBoolean GreaterThan(SqlDateTime x, SqlDateTime y);
23 [C++] public: static SqlBoolean GreaterThan(SqlDateTime x, SqlDateTime y);
24 [VB] Public Shared Function GreaterThan(ByVal x As SqlDateTime, ByVal y As
25 SqlDateTime) As SqlBoolean

```

1 [JScript] public static function GreaterThan(x : SqlDateTime, y : SqlDateTime) :  
2 SqlBoolean;

3  
4 *Description*

5 [ .]

6 GreaterThanOrEqualTo

7  
8 [C#] public static SqlBoolean GreaterThanOrEqualTo(SqlDateTime x, SqlDateTime  
9 y);

10 [C++] public: static SqlBoolean GreaterThanOrEqualTo(SqlDateTime x,  
11 SqlDateTime y);

12 [VB] Public Shared Function GreaterThanOrEqualTo(ByVal x As SqlDateTime,  
13 ByVal y As SqlDateTime) As SqlBoolean

14 [JScript] public static function GreaterThanOrEqualTo(x : SqlDateTime, y :  
15 SqlDateTime) : SqlBoolean;

16  
17 *Description*

18 [ .]

19 LessThan

20  
21 [C#] public static SqlBoolean LessThan(SqlDateTime x, SqlDateTime y);

22 [C++] public: static SqlBoolean LessThan(SqlDateTime x, SqlDateTime y);

23 [VB] Public Shared Function LessThan(ByVal x As SqlDateTime, ByVal y As  
24 SqlDateTime) As SqlBoolean

25 [JScript] public static function LessThan(x : SqlDateTime, y : SqlDateTime) :

1 SqlBoolean;

2

3 *Description*

4 [ .]

5 LessThanOrEqualTo

6

7 [C#] public static SqlBoolean LessThanOrEqualTo(SqlDateTime x, SqlDateTime y);

8 [C++] public: static SqlBoolean LessThanOrEqualTo(SqlDateTime x, SqlDateTime

9 y);

10 [VB] Public Shared Function LessThanOrEqualTo(ByVal x As SqlDateTime, ByVal

11 y As SqlDateTime) As SqlBoolean

12 [JScript] public static function LessThanOrEqualTo(x : SqlDateTime, y :

13 SqlDateTime) : SqlBoolean;

14

15 *Description*

16 [ .]

17 NotEquals

18

19 [C#] public static SqlBoolean NotEquals(SqlDateTime x, SqlDateTime y);

20 [C++] public: static SqlBoolean NotEquals(SqlDateTime x, SqlDateTime y);

21 [VB] Public Shared Function NotEquals(ByVal x As SqlDateTime, ByVal y As

22 SqlDateTime) As SqlBoolean

23 [JScript] public static function NotEquals(x : SqlDateTime, y : SqlDateTime) :

24 SqlBoolean;

25

*Description*

[ .]

op\_Addition

[C#] public static SqlDateTime operator +(SqlDateTime x, TimeSpan t);

[C++] public: static SqlDateTime op\_Addition(SqlDateTime x, TimeSpan t);

[VB]       returnValue       =       SqlDateTime.op\_Addition(x,       t)

[JScript]       returnValue       =       x       +       t;

*Description*

Adds the amount of time indicated by the supplied TimeSpan parameter, *t*, to the supplied **System.Data.SqlTypes.SqlDateTime** structure.

**Return Value:** A new **System.Data.SqlTypes.SqlDateTime**. If either argument is **System.Data.SqlTypes.SqlDateTime.Null**, the new **System.Data.SqlTypes.SqlDateTime.Value** will be **System.Data.SqlTypes.SqlDateTime.Null**. A **System.Data.SqlTypes.SqlDateTime** structure. A **System.TimeSpan** structure.

op\_Equality

[C#] public static SqlBoolean operator ==(SqlDateTime x, SqlDateTime y);

[C++] public: static SqlBoolean op\_Equality(SqlDateTime x, SqlDateTime y);

[VB]       returnValue       =       SqlDateTime.op\_Equality(x,       y)

[JScript]       returnValue       =       x       ==       y;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Performs a logical comparison of two **System.Data.SqlTypes.SqlDateTime** structures to determine if they are equal.  
*Return Value:* **true** if the two values are equal, otherwise **false**. A **System.Data.SqlTypes.SqlDateTime** structure. A **System.Data.SqlTypes.SqlDateTime** structure.

op\_Explicit

```
[C#] public static explicit operator DateTime(SqlDateTime x);  
[C++] public: static DateTime op_Explicit();  
[VB] returnValue = SqlDateTime.op_Explicit(x)  
[JScript] returnValue = DateTime(x);
```

*Description*

Converts a **System.Data.SqlTypes.SqlDateTime** structure to a **System.DateTime** structure.  
*Return Value:* A **System.DateTime** object whose **System.DateTime.Date** and **System.TimeOfDay** properties contain the same date and time values as the **System.Data.SqlTypes.SqlDateTime.Value** property of the supplied **System.Data.SqlTypes.SqlDateTime** structure. A **System.Data.SqlTypes.SqlDateTime** structure.

op\_Explicit

```
[C#] public static explicit operator SqlDateTime(SqlString x);
```

```

1  [C++]    public:    static    SqlDateTime    op_Explicit(SqlString    x);
2  [VB]      returnValue    =    SqlDateTime.op_Explicit(x)
3  [JScript]      returnValue    =    SqlDateTime(x);

```

#### 5 *Description*

6 Converts the supplied **System.Data.SqlTypes.SqlString** to a  
7 **System.Data.SqlTypes.SqlDateTime** structure.

8 *Return Value:* A **System.Data.SqlTypes.SqlDateTime** structure whose  
9 **System.Data.SqlTypes.SqlDateTime.Value** is equal to the date and time  
10 represented by the **System.Data.SqlTypes.SqlString** parameter. If the  
11 **System.Data.SqlTypes.SqlString** is null, the  
12 **System.Data.SqlTypes.SqlDateTime.Value** of the newly created  
13 **System.Data.SqlTypes.SqlDateTime** structure will be null. A  
14 **System.Data.SqlTypes.SqlString** to be converted.

15 op\_GreaterThan

```

17 [C#] public static SqlBoolean operator >(SqlDateTime x, SqlDateTime y);
18 [C++] public: static SqlBoolean op_GreaterThan(SqlDateTime x, SqlDateTime y);
19 [VB]      returnValue    =    SqlDateTime.op_GreaterThan(x,    y)
20 [JScript]      returnValue    =    x    >    y;

```

#### 22 *Description*

23 Compares two instances of **System.Data.SqlTypes.SqlDateTime** to  
24 determine if the first is greater than the second.

25 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is

1 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the  
 2 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
 3 instance of **System.Data.SqlTypes.SqlByte** is null, the  
 4 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 5 **System.Data.SqlTypes.SqlBoolean** will be  
 6 **System.Data.SqlTypes.SqlBoolean.Null** . A  
 7 **System.Data.SqlTypes.SqlDateTime** structure. A  
 8 **System.Data.SqlTypes.SqlDateTime** structure.

9 **op\_GreaterThanOrEqual**  
 10  
 11 [C#] public static SqlBoolean operator >=(SqlDateTime x, SqlDateTime y);  
 12 [C++] public: static SqlBoolean op\_GreaterThanOrEqual(SqlDateTime x,  
 13 SqlDateTime y);  
 14 [VB] returnValue = SqlDateTime.op\_GreaterThanOrEqual(x, y)  
 15 [JScript] returnValue = x >= y;  
 16

### 17 *Description*

18 Compares two instances of **System.Data.SqlTypes.SqlDateTime** to  
 19 determine if the first is greater than or equal to the second.

20 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 21 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greaater than or  
 22 equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**  
 23 . If either instance of **System.Data.SqlTypes.SqlDateTime** is null, the  
 24 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 25 **System.Data.SqlTypes.SqlBoolean** will be

1 **System.Data.SqlTypes.SqlBoolean.Null** A

2 **System.Data.SqlTypes.SqlDateTime** structure. A

3 **System.Data.SqlTypes.SqlDateTime** structure.

4 **op\_Implicit**

6 [C#] public static implicit operator SqlDateTime(DateTime value);

7 [C++] public: static SqlDateTime op\_Implicit(DateTime value);

8 [VB] returnValue = SqlDateTime.op\_Implicit(value)

9 [JScript] returnValue = value;

11 *Description*

12 Converts a **System.DateTime** structure to a  
13 **System.Data.SqlTypes.SqlDateTime** structure.

14 *Return Value:* A **System.Data.SqlTypes.SqlDateTime** structure whose  
15 **System.Data.SqlTypes.SqlDateTime.Value** is equal to the combined  
16 **System.DateTime.Date** and **System.TimeOfDay** properties of the supplied  
17 **System.DateTime** structure. A **System.DateTime** structure.

18 **op\_Inequality**

20 [C#] public static SqlBoolean operator !=(SqlDateTime x, SqlDateTime y);

21 [C++] public: static SqlBoolean op\_Inequality(SqlDateTime x, SqlDateTime y);

22 [VB] returnValue = SqlDateTime.op\_Inequality(x, y)

23 [JScript] returnValue = x != y;

25 *Description*

Performs a logical comparison of two instances of **System.Data.SqlTypes.SqlDateTime** to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **System.Data.SqlTypes.SqlDateTime** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlDateTime** structure. A **System.Data.SqlTypes.SqlDateTime** structure.

**op\_LessThan**

[C#]            public            static            SqlBoolean            operator

[C++] public: static SqlBoolean op\_LessThan(SqlDateTime x, SqlDateTime y);

[VB]            returnValue            =            SqlDateTime.op\_LessThan(x,            y)

[JScript]            returnValue            =            x            <            y;

### *Description*

Compares two instances of **System.Data.SqlTypes.SqlDateTime** to determine if the first is less than the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlDateTime** is null, the

1 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 2 **System.Data.SqlTypes.SqlBoolean** will be  
 3 **System.Data.SqlTypes.SqlBoolean.Null** . A  
 4 **System.Data.SqlTypes.SqlDateTime** structure. A  
 5 **System.Data.SqlTypes.SqlDateTime** structure.

6 **op\_LessThanOrEqual**

7  
 8 [C#] public static SqlBoolean operator <=(SqlDateTime x, SqlDateTime y);  
 9 [C++] public: static SqlBoolean op\_LessThanOrEqual(SqlDateTime x,  
 10 SqlDateTime y);  
 11 [VB] returnValue = SqlDateTime.op\_LessThanOrEqual(x, y)  
 12 [JScript] returnValue = x <= y;

13  
 14 *Description*

15 Compares two instances of **System.Data.SqlTypes.SqlDateTime** to  
 16 determine if the first is less than or equal to the second.

17 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 18 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal  
 19 to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If

20 either instance of **System.Data.SqlTypes.SqlDateTime** is null, the

21 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 22 **System.Data.SqlTypes.SqlBoolean** will be  
 23 **System.Data.SqlTypes.SqlBoolean.Null** . A  
 24 **System.Data.SqlTypes.SqlDateTime** structure. A  
 25 **System.Data.SqlTypes.SqlDateTime** structure.





Represents a fixed precision and scale numeric value between -10 -1 and 10  
-1 to be stored in or retrieved from a database.

ToString

[C#]	public	static	readonly	byte	MaxPrecision;
[C++]	public:	static	unsigned	char	MaxPrecision;
[VB]	Public	Shared	ReadOnly	MaxPrecision	As Byte
[JScript]	public	static	var	MaxPrecision	: Byte;

#### Description

A constant representing the largest possible value for the  
**System.Data.SqlTypes.SqlDecimal.Precision** property.

The value of this constant is 38.

ToString

[C#]	public	static	readonly	byte	MaxScale;
[C++]	public:	static	unsigned	char	MaxScale;
[VB]	Public	Shared	ReadOnly	MaxScale	As Byte
[JScript]	public	static	var	MaxScale	: Byte;

#### Description

A constant representing the maximum value for the  
**System.Data.SqlTypes.SqlDecimal.Scale** property.

ToString



```

1  [C++]          public:          static          SqlDecimal          Null;
2  [VB]   Public   Shared   ReadOnly   Null   As   SqlDecimal
3  [JScript]   public   static   var   Null   :   SqlDecimal;

```

#### 5 *Description*

6 Represents a null value that can be assigned to the  
7 **System.Data.SqlTypes.SqlDecimal.Value** property of an instance of the  
8 **System.Data.SqlTypes.SqlMoney** class.

9 **SqlDecimal**

#### 10 *Example Syntax:*

11 **ToString**

```

13 [C#]          public          SqlDecimal(decimal          value);
14 [C++]          public:          SqlDecimal(Decimal          value);
15 [VB]   Public   Sub   New(ByVal   value   As   Decimal)
16 [JScript] public function SqlDecimal(value : Decimal); Initializes a new instance
17 of          the          System.Data.SqlTypes.SqlDecimal          structure.

```

#### 19 *Description*

20 Initializes a new instance of the **System.Data.SqlTypes.SqlDecimal**  
21 structure using the supplied **System.Decimal** value. The **System.Decimal** value to  
22 be stored as a **System.Data.SqlTypes.SqlDecimal** structure.

23 **SqlDecimal**

#### 24 *Example Syntax:*

25 **ToString**

```

1
2 [C#]          public          SqlDecimal(double          dVal);
3 [C++]          public:          SqlDecimal(double          dVal);
4 [VB]      Public      Sub      New(ByVal      dVal      As      Double)
5 [JScript]      public      function      SqlDecimal(dVal      :      double);
6

```

#### 7 *Description*

8        Initializes a new instance of the **System.Data.SqlTypes.SqlDecimal**  
9 structure using the supplied double parameter. A double, representing the value for  
10 the new **System.Data.SqlTypes.SqlDecimal** structure.

11        SqlDecimal

12        *Example Syntax:*

13        ToString

```

14
15 [C#]          public          SqlDecimal(int          value);
16 [C++]          public:          SqlDecimal(int          value);
17 [VB]      Public      Sub      New(ByVal      value      As      Integer)
18 [JScript]      public      function      SqlDecimal(value      :      int);
19

```

#### 20 *Description*

21        Initializes a new instance of the **System.Data.SqlTypes.SqlDecimal**  
22 structure using the supplied integer value. The supplied integer value which will  
23 be used as the value of the new **System.Data.SqlTypes.SqlDecimal** structure.

24        SqlDecimal

25        *Example Syntax:*

ToString

```
[C#]          public          SqlDecimal(long          value);
[C++]          public:          SqlDecimal(__int64          value);
[VB]      Public      Sub      New(ByVal      value      As      Long)
[JScript]      public      function      SqlDecimal(value      :      long);
```

### Description

Initializes a new instance of the **System.Data.SqlTypes.SqlDecimal** structure using the supplied long integer value. The supplied long integer value which will be used as the value of the new **System.Data.SqlTypes.SqlDecimal** structure.

SqlDecimal

*Example Syntax:*

ToString

```
[C#] public SqlDecimal(byte bPrecision, byte bScale, bool fPositive, int[] bits);
[C++] public: SqlDecimal(unsigned char bPrecision, unsigned char bScale, bool
fPositive,          int          bits          __gc[]);
[VB] Public Sub New(ByVal bPrecision As Byte, ByVal bScale As Byte, ByVal
fPositive      As      Boolean,      ByVal      bits()      As      Integer)
[JScript] public function SqlDecimal(bPrecision : Byte, bScale : Byte, fPositive :
Boolean,          bits          :          int[]);
```

### Description

1        Initializes a new instance of the **System.Data.SqlTypes.SqlDecimal**  
 2 structure using the supplied parameters. The maximum number of digits that can  
 3 be used to represent the **System.Data.SqlTypes.SqlDecimal.Value** property of  
 4 the new **System.Data.SqlTypes.SqlDecimal** structure. The number of decimal  
 5 places to which the **System.Data.SqlTypes.SqlDecimal.Value** property will be  
 6 resolved for the new **System.Data.SqlTypes.SqlDecimal** structure. [ .]

7        **SqlDecimal**

8        *Example Syntax:*

9        **ToString**

11    [C#] public SqlDecimal(byte bPrecision, byte bScale, bool fPositive, int data1, int  
 12 data2,                    int                    data3,                    int                    data4);

13    [C++] public: SqlDecimal(unsigned char bPrecision, unsigned char bScale, bool  
 14 fPositive,    int    data1,    int    data2,    int    data3,    int    data4);

15    [VB] Public Sub New(ByVal bPrecision As Byte, ByVal bScale As Byte, ByVal  
 16 fPositive As Boolean, ByVal data1 As Integer, ByVal data2 As Integer, ByVal  
 17 data3        As        Integer,        ByVal        data4        As        Integer)

18    [JScript] public function SqlDecimal(bPrecision : Byte, bScale : Byte, fPositive :  
 19 Boolean,    data1    : int,    data2    : int,    data3    : int,    data4    : int);

21    *Description*

22        Initializes a new instance of the **System.Data.SqlTypes.SqlDecimal**  
 23 structure using the supplied parameters. The maximum number of digits that can  
 24 be used to represent the **System.Data.SqlTypes.SqlDecimal.Value** property of  
 25 the new **System.Data.SqlTypes.SqlDecimal** structure. The number of decimal

1 places to which the **System.Data.SqlTypes.SqlDecimal.Value** property will be  
 2 resolved for the new **System.Data.SqlTypes.SqlDecimal** structure. [ .][ .][ .]

3 [ .][ .]

4 BinData

5 ToString

6  
 7 [C#] public byte[] BinData {get;}

8 [C++] public: \_\_property unsigned char get\_BinData();

9 [VB] Public ReadOnly Property BinData As Byte ()

10 [JScript] public function get BinData() : Byte[];

11  
 12 *Description*

13 [ .][ .]

14 Data

15 ToString

16  
 17 [C#] public int[] Data {get;}

18 [C++] public: \_\_property int get\_Data();

19 [VB] Public ReadOnly Property Data As Integer ()

20 [JScript] public function get Data() : int[];

21  
 22 *Description*

23 [ .][ .]

24 IsNull

25 ToString

2	[C#]	public		bool	IsNull		{get;}
3	[C++]	public:		__property	bool		get_IsNull();
4	[VB]	Public	ReadOnly	Property	IsNull	As	Boolean
5	[JScript]	public	function	get	IsNull()	:	Boolean;

### Description

Indicates whether or not the **System.Data.SqlTypes.SqlDecimal.Value** of this **System.Data.SqlTypes.SqlDecimal** structure is null.

## IsPositive

ToString

13	[C#]	public		bool	IsPositive		{get;}
14	[C++]	public:		__property	bool		get_IsPositive();
15	[VB]	Public	ReadOnly	Property	IsPositive	As	Boolean
16	[JScript]	public	function	get	IsPositive()	:	Boolean;

### Description

Indicates whether or not the **System.Data.SqlTypes.SqlDecimal.Value** of this **System.Data.SqlTypes.SqlDecimal** structure is greater than zero.

## Precision

ToString

24	[C#]	public	byte	Precision	{get;}
25	[C++]	public:	property	unsigned char	get Precision();

```

1  [VB]      Public      ReadOnly      Property      Precision      As      Byte
2  [JScript]      public      function      get      Precision()      :      Byte;

```

```

3
4  Description

```

```

5      Gets or sets the maximum number of digits used to represent the
6  System.Data.SqlTypes.SqlDecimal.Value property.

```

```

7      Scale
8      ToString

```

```

9
10 [C#]          public          byte          Scale          {get;}
11 [C++]      public:      __property      unsigned      char      get_Scale();
12 [VB]      Public      ReadOnly      Property      Scale      As      Byte
13 [JScript]      public      function      get      Scale()      :      Byte;

```

```

14
15 Description
16      Gets or sets the number of decimal places to which
17 System.Data.SqlTypes.SqlDecimal.Value is resolved.

```

```

18      Value
19      ToString

```

```

20
21 [C#]          public          decimal          Value          {get;}
22 [C++]      public:      __property      Decimal          get_Value();
23 [VB]      Public      ReadOnly      Property      Value      As      Decimal
24 [JScript]      public      function      get      Value()      :      Decimal;

```

```

25

```

## Description

Gets the value of the **System.Data.SqlTypes.SqlDecimal** structure. This property is read-only.

### Abs

```
[C#]      public      static      SqlDecimal      Abs(SqlDecimal      n);
[C++]     public:     static      SqlDecimal      Abs(SqlDecimal      n);
[VB]     Public Shared Function Abs(ByVal n As SqlDecimal) As SqlDecimal
[JScript] public static function Abs(n : SqlDecimal) : SqlDecimal;
```

## Description

The **Abs** member function gets the absolute value of the **System.Data.SqlTypes.SqlDecimal** parameter.

**Return Value:** A **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** property contains the unsigned number representing the absolute value of the **System.Data.SqlTypes.SqlDecimal** parameter. A **SqlDecimal** structure.

### Add

```
[C#]      public      static      SqlDecimal      Add(SqlDecimal      x,      SqlDecimal      y);
[C++]     public:     static      SqlDecimal      Add(SqlDecimal      x,      SqlDecimal      y);
[VB]     Public Shared Function Add(ByVal x As SqlDecimal, ByVal y As
SqlDecimal)
As
SqlDecimal
[JScript] public static function Add(x : SqlDecimal, y : SqlDecimal) : SqlDecimal;
```

## Description

[ .]

### AdjustScale

[C#] public static SqlDecimal AdjustScale(SqlDecimal n, int digits, bool fRound);

[C++] public: static SqlDecimal AdjustScale(SqlDecimal n, int digits, bool fRound);

[VB] Public Shared Function AdjustScale(ByVal n As SqlDecimal, ByVal digits As Integer, ByVal fRound As Boolean) As SqlDecimal

[JScript] public static function AdjustScale(n : SqlDecimal, digits : int, fRound : Boolean) : SqlDecimal;

## Description

The scale of the **System.Data.SqlTypes.SqlDecimal** operand will be adjusted to the number of digits indicated by the digits parameter. Depending on the value of the fRound parameter, the value will either be rounded to the appropriate number of digits or truncated.

**Return Value:** A new **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** property contains the adjusted number. The SqlDecimal structure to be adjusted. The number of digits in the adjusted structure. If this parameter is **true**, the new Value will be rounded, if **false**, the value will be truncated.

### Ceiling

```

1
2 [C#]      public      static      SqlDecimal      Ceiling(SqlDecimal      n);
3 [C++]     public:     static      SqlDecimal      Ceiling(SqlDecimal      n);
4 [VB] Public Shared Function Ceiling(ByVal n As SqlDecimal) As SqlDecimal
5 [JScript] public static function Ceiling(n : SqlDecimal) : SqlDecimal;
6

```

#### *Description*

```

8      [ .][ .][ .]
9      CompareTo
10

```

```

11 [C#]      public      int      CompareTo(object      value);
12 [C++]     public:     __sealed  int      CompareTo(Object*      value);
13 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
14 Integer
15 [JScript] public function CompareTo(value : Object) : int;
16

```

#### *Description*

Compares this instance to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

```

22      ConvertToPrecScale
23

```

```

24 [C#] public static SqlDecimal ConvertToPrecScale(SqlDecimal n, int precision,
25 int scale);

```

```

1 [C++] public: static SqlDecimal ConvertToPrecScale(SqlDecimal n, int precision,
2 int scale);
3 [VB] Public Shared Function ConvertToPrecScale(ByVal n As SqlDecimal,
4 ByVal precision As Integer, ByVal scale As Integer) As SqlDecimal
5 [JScript] public static function ConvertToPrecScale(n : SqlDecimal, precision :
6 int, scale : int) : SqlDecimal;

```

### 8 *Description*

9 Adjusts the value of the **System.Data.SqlTypes.SqlDecimal** operand to the  
10 indicated precision and scale.

11 *Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose Value  
12 has been adjusted to the precision and scale indicated in the parameters. The  
13 SqlDecimal structure whose value is to be adjusted. The precision for the new  
14 SqlDecimal structure. The scale for the new SqlDecimal structure.

### 15 *Divide*

```

17 [C#] public static SqlDecimal Divide(SqlDecimal x, SqlDecimal y);
18 [C++] public: static SqlDecimal Divide(SqlDecimal x, SqlDecimal y);
19 [VB] Public Shared Function Divide(ByVal x As SqlDecimal, ByVal y As
20 SqlDecimal) As SqlDecimal
21 [JScript] public static function Divide(x : SqlDecimal, y : SqlDecimal) :
22 SqlDecimal;

```

### 24 *Description*

25 [ .]

## Equals

```
[C#]      public      override      bool      Equals(object      value);  
[C++]      public:      bool      Equals(Object*      value);  
[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean  
[JScript] public override function Equals(value : Object) : Boolean;
```

### *Description*

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlMoney.Value** property of the **System.Data.SqlTypes.SqlMoney** object. The object to be compared.

## Equals

```
[C#] public static new SqlBoolean Equals(SqlDecimal x, SqlDecimal y);  
[C++] public: static SqlBoolean Equals(SqlDecimal x, SqlDecimal y);  
[VB] Shadows Public Shared Function Equals(ByVal x As SqlDecimal, ByVal y  
As SqlDecimal) As SqlBoolean  
[JScript] public static hide function Equals(x : SqlDecimal, y : SqlDecimal) :  
SqlBoolean;
```

### *Description*

[ . ]

## Floor

```
[C#]      public      static      SqlDecimal      Floor(SqlDecimal      n);
```

```

1  [C++]      public:      static      SqlDecimal      Floor(SqlDecimal      n);
2  [VB] Public Shared Function Floor(ByVal n As SqlDecimal) As SqlDecimal
3  [JScript] public static function Floor(n : SqlDecimal) : SqlDecimal;

```

4  
5 *Description*

6 [ . ][ . ][ . ]  
7 GetHashCode

```

8
9  [C#]      public      override      int      GetHashCode();
10 [C++]      public:      int      GetHashCode();
11 [VB] Overrides Public Function GetHashCode() As Integer
12 [JScript] public override function GetHashCode() : int;

```

13  
14 *Description*

15 Returns the hash code for this instance.

16 *Return Value:* A 32-bit signed integer hash code.

17 GreaterThan

```

18
19 [C#] public static SqlBoolean GreaterThan(SqlDecimal x, SqlDecimal y);
20 [C++] public: static SqlBoolean GreaterThan(SqlDecimal x, SqlDecimal y);
21 [VB] Public Shared Function GreaterThan(ByVal x As SqlDecimal, ByVal y As
22 SqlDecimal) As SqlBoolean
23 [JScript] public static function GreaterThan(x : SqlDecimal, y : SqlDecimal) :
24 SqlBoolean;

```

25

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

[ .]

GreaterThanOrEqualTo

```
[C#] public static SqlBoolean GreaterThanOrEqualTo(SqlDecimal x, SqlDecimal y);
[C++] public: static SqlBoolean GreaterThanOrEqualTo(SqlDecimal x, SqlDecimal
y);
[VB] Public Shared Function GreaterThanOrEqualTo(ByVal x As SqlDecimal,
ByVal y As SqlDecimal) As SqlBoolean
[JScript] public static function GreaterThanOrEqualTo(x : SqlDecimal, y :
SqlDecimal) : SqlBoolean;
```

*Description*

[ .]

LessThan

```
[C#] public static SqlBoolean LessThan(SqlDecimal x, SqlDecimal y);
[C++] public: static SqlBoolean LessThan(SqlDecimal x, SqlDecimal y);
[VB] Public Shared Function LessThan(ByVal x As SqlDecimal, ByVal y As
SqlDecimal) As SqlBoolean
[JScript] public static function LessThan(x : SqlDecimal, y : SqlDecimal) :
SqlBoolean;
```

*Description*

```

1      [ .]
2      LessThanOrEqualTo
3
4      [C#] public static SqlBoolean LessThanOrEqualTo(SqlDecimal x, SqlDecimal y);
5      [C++] public: static SqlBoolean LessThanOrEqualTo(SqlDecimal x, SqlDecimal y);
6      [VB] Public Shared Function LessThanOrEqualTo(ByVal x As SqlDecimal, ByVal y
7      As          SqlDecimal)          As          SqlBoolean
8      [JScript] public static function LessThanOrEqualTo(x : SqlDecimal, y : SqlDecimal)
9      :          SqlBoolean;

```

11 *Description*

```

12      [ .]
13      Multiply
14
15      [C#] public static SqlDecimal Multiply(SqlDecimal x, SqlDecimal y);
16      [C++] public: static SqlDecimal Multiply(SqlDecimal x, SqlDecimal y);
17      [VB] Public Shared Function Multiply(ByVal x As SqlDecimal, ByVal y As
18      SqlDecimal)          As          SqlDecimal
19      [JScript] public static function Multiply(x : SqlDecimal, y : SqlDecimal) :
20      SqlDecimal;

```

22 *Description*

```

23      [ .]
24      NotEquals

```

25

```

1
2 [C#] public static SqlBoolean NotEquals(SqlDecimal x, SqlDecimal y);
3 [C++] public: static SqlBoolean NotEquals(SqlDecimal x, SqlDecimal y);
4 [VB] Public Shared Function NotEquals(ByVal x As SqlDecimal, ByVal y As
5 SqlDecimal) As SqlBoolean
6 [JScript] public static function NotEquals(x : SqlDecimal, y : SqlDecimal) :
7 SqlBoolean;
8

```

9 *Description*

```

10 [ .]
11 op_Addition
12

```

```

13 [C#] public static SqlDecimal operator +(SqlDecimal x, SqlDecimal y);
14 [C++] public: static SqlDecimal op_Addition(SqlDecimal x, SqlDecimal y);
15 [VB] returnValue = SqlDecimal.op_Addition(x, y)
16 [JScript] returnValue = x + y;
17

```

18 *Description*

19 The addition operator calculates the sum of the two  
20 **System.Data.SqlTypes.SqlDecimal** operators.

21 *Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose  
22 **System.Data.SqlTypes.SqlDecimal.Value** property contains the sum. A  
23 **System.Data.SqlTypes.SqlDecimal** structure. A  
24 **System.Data.SqlTypes.SqlDecimal** structure.

25 op\_Division

```

1
2 [C#] public static SqlDecimal operator /(SqlDecimal x, SqlDecimal y);
3 [C++] public: static SqlDecimal op_Division(SqlDecimal x, SqlDecimal y);
4 [VB]      returnValue      =      SqlDecimal.op_Division(x,      y)
5 [JScript]      returnValue      =      x      /      y;

```

#### Description

The division operator calculates the results of dividing the first **System.Data.SqlTypes.SqlDecimal** operand by the second.

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** property contains the results of the division. A **System.Data.SqlTypes.SqlDecimal** structure. A **System.Data.SqlTypes.SqlDecimal** structure.

#### op\_Equality

```

16 [C#] public static SqlBoolean operator ==(SqlDecimal x, SqlDecimal y);
17 [C++] public: static SqlBoolean op_Equality(SqlDecimal x, SqlDecimal y);
18 [VB]      returnValue      =      SqlDecimal.op_Equality(x,      y)
19 [JScript]      returnValue      =      x      ==      y;

```

#### Description

Performs a logical comparison of the two **System.Data.SqlTypes.SqlDecimal** operands to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or

1 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If  
 2 either instance of **System.Data.SqlTypes.SqlDecimal** is null, the  
 3 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 4 **System.Data.SqlTypes.SqlBoolean** will be  
 5 **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlDecimal**  
 6 structure. A **System.Data.SqlTypes.SqlDecimal** structure.

7 **op\_Explicit**

8  
 9 [C#] public static explicit operator SqlDecimal(SqlBoolean x);  
 10 [C++] public: static SqlDecimal op\_Explicit(SqlBoolean x);  
 11 [VB] returnValue = SqlDecimal.op\_Explicit(x)  
 12 [JScript] returnValue = SqlDecimal(x);

13  
 14 *Description*

15 Converts the supplied **System.Data.SqlTypes.SqlBit** structure to  
 16 **System.Data.SqlTypes.SqlDecimal**.  
 17 *Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose  
 18 **System.Data.SqlTypes.SqlDecimal.Value** is equal to the  
 19 **System.Data.SqlTypes.SqlBit.ByteValue** of the **System.Data.SqlTypes.SqlBit**  
 20 parameter. The **System.Data.SqlTypes.SqlBit** structure to be converted.

21 **op\_Explicit**

22  
 23 [C#] public static explicit operator decimal(SqlDecimal x);  
 24 [C++] public: static Decimal op\_Explicit();  
 25 [VB] returnValue = SqlDecimal.op\_Explicit(x)

1 [JScript]                      returnValue                      =                      Decimal(x);

2

3 *Description*

4            Converts    the    **System.Data.SqlTypes.SqlDecimal**    parameter    to  
5 **System.Decimal**

6 *Return Value:* A new **System.Decimal** structure whose value equals the  
7 **System.Data.SqlTypes.SqlDecimal.Value**                      of                      the  
8 **System.Data.SqlTypes.SqlDecimal**                      parameter.                      The  
9 **System.Data.SqlTypes.SqlDecimal** structure to be converted.

10            op\_Explicit

11

12 [C#]    public    static    explicit    operator    SqlDecimal(SqlDouble    x);

13 [C++]    public:    static    SqlDecimal    op\_Explicit(SqlDouble    x);

14 [VB]            returnValue            =            SqlDecimal.op\_Explicit(x)

15 [JScript]            returnValue            =            SqlDecimal(x);

16

17 *Description*

18            Converts the supplied **System.Data.SqlTypes.SqlDouble** structure to  
19 **System.Data.SqlTypes.SqlDecimal**

20 *Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose  
21 **System.Data.SqlTypes.SqlDecimal.Value**                      equals                      the  
22 **System.Data.SqlTypes.SqlDouble.Value**                      of                      the  
23 **System.Data.SqlTypes.SqlDouble**                      parameter.                      The  
24 **System.Data.SqlTypes.SqlDouble** structure to be converted.

25            op\_Explicit

2	[C#]	public	static	explicit	operator	SqlDecimal(SqlSingle x);
3	[C++]	public:	static		SqlDecimal	op_Explicit(SqlSingle x);
4	[VB]		returnValue		=	SqlDecimal.op_Explicit(x)
5	[JScript]		returnValue		=	SqlDecimal(x);

### Description

Converts the supplied **System.Data.SqlTypes.SqlSingle** structure to **System.Data.SqlTypes.SqlDecimal**.

**Return Value:** A new **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** property equals the **System.Data.SqlTypes.SqlSingle.Value** of the **System.Data.SqlTypes.SqlSingle** parameter. The **System.Data.SqlTypes.SqlSingle** structure to be converted.

op\_Explicit

16	[C#]	public	static	explicit	operator	SqlDecimal(SqlString x);
17	[C++]	public:	static	SqlDecimal	op_Explicit(SqlString x);	
18	[VB]		returnValue	=		SqlDecimal.op_Explicit(x)
19	[JScript]		returnValue	=		SqlDecimal(x);

### Description

Converts the supplied **System.Data.SqlTypes.SqlString** parameter to **System.Data.SqlTypes.SqlDecimal**.

**Return Value:** A new `System.Data.SqlTypes.SqlDecimal` structure whose `System.Data.SqlTypes.SqlDecimal.Value` equals the value represented by the

**System.Data.SqlTypes.SqlString** parameter. The

**System.Data.SqlTypes.SqlString** object to be converted.

**op\_GreaterThan**

[C#] public static SqlBoolean operator >(SqlDecimal x, SqlDecimal y);

[C++] public: static SqlBoolean op\_GreaterThan(SqlDecimal x, SqlDecimal y);

[VB] returnValue = SqlDecimal.op\_GreaterThan(x, y)

[JScript] returnValue = x > y;

### *Description*

Performs a logical comparison of two **System.Data.SqlTypes.SqlDecimal** structures to determine if the first is greater than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlDecimal** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlDecimal** structure. A **System.Data.SqlTypes.SqlDecimal** structure.

**op\_GreaterThanOrEqual**

[C#] public static SqlBoolean operator >=(SqlDecimal x, SqlDecimal y);

[C++] public: static SqlBoolean op\_GreaterThanOrEqual(SqlDecimal x,

SqlDecimal y);

1 [VB]      returnValue      =      SqlDecimal.op\_GreaterThanOrEqual(x,      y)  
 2 [JScript]      returnValue      =      x      >=      y;

3

4 *Description*

5      Performs      a      logical      comparison      of      the      two  
 6 **System.Data.SqlTypes.SqlDecimal** parameters to determine if the first is greater  
 7 than      or      equal      to      the      second.

8 *Return Value:*      A      **System.Data.SqlTypes.SqlBoolean**      that      is  
 9 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or  
 10 equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**  
 11 . If either instance of **System.Data.SqlTypes.SqlDecimal** is null, the  
 12 **System.Data.SqlTypes.SqlBoolean.Value**      of      the  
 13 **System.Data.SqlTypes.SqlBoolean**      will      be  
 14 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlDecimal**  
 15 structure. A **System.Data.SqlTypes.SqlDecimal** structure.

16      op\_Explicit

17

18 [C#]      public      static      implicit      operator      SqlDecimal(decimal      x);  
 19 [C++]      public:      static      SqlDecimal      op\_Explicit(Decimal      x);  
 20 [VB]           returnValue      =      SqlDecimal.op\_Explicit(x)  
 21 [JScript]           returnValue      =      x;

22

23 *Description*

24      Converts      the      **System.Decimal**      value      to  
 25 **System.Data.SqlTypes.SqlDecimal** .

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** property equals the value of the **System.Decimal** parameter. The decimal value to be converted.

op\_Implicit

```
[C#]    public    static    implicit    operator    SqlDecimal(SqlByte    x);
[C++]    public:    static    SqlDecimal    op_Implicit(SqlByte    x);
[VB]        returnValue    =    SqlDecimal.op_Implicit(x)
[JScript]        returnValue    =    x;
```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlByte** structure to **System.Data.SqlTypes.SqlDecimal**. The **System.Data.SqlTypes.SqlByte** structure to be converted.

op\_Implicit

```
[C#]    public    static    implicit    operator    SqlDecimal(SqlInt16    x);
[C++]    public:    static    SqlDecimal    op_Implicit(SqlInt16    x);
[VB]        returnValue    =    SqlDecimal.op_Implicit(x)
[JScript]        returnValue    =    x;
```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlInt16** structure to **System.Data.SqlTypes.SqlDecimal**

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose

1 **System.Data.SqlTypes.SqlDecimal.Value** property equals the  
 2 **System.Data.SqlTypes.SqlInt16.Value** property of the  
 3 **System.Data.SqlTypes.SqlInt16** parameter. The  
 4 **System.Data.SqlTypes.SqlInt16** structure to be converted.

5 **op\_Implicit**

6  
 7 [C#] public static implicit operator SqlDecimal(SqlInt32 x);  
 8 [C++] public: static SqlDecimal op\_Implicit(SqlInt32 x);  
 9 [VB] returnValue = SqlDecimal.op\_Implicit(x)  
 10 [JScript] returnValue = x;

# 11 *Description*

12 Converts the supplied **System.Data.SqlTypes.SqlInt32** structure to  
 13 **System.Data.SqlTypes.SqlDecimal**.  
 14 *Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose  
 15 **System.Data.SqlTypes.Value** property is equal to the  
 16 **System.Data.SqlTypes.Value** property of the **System.Data.SqlTypes.SqlInt32**  
 17 parameter. The **System.Data.SqlTypes.SqlInt32** structure to be converted.  
 18

19 **op\_Implicit**

20  
 21 [C#] public static implicit operator SqlDecimal(SqlInt64 x);  
 22 [C++] public: static SqlDecimal op\_Implicit(SqlInt64 x);  
 23 [VB] returnValue = SqlDecimal.op\_Implicit(x)  
 24 [JScript] returnValue = x;

25

*Description*

Converts the supplied **System.Data.SqlTypes.SqlInt64** structure to **SqlDecimal**.

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** equals the **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64** parameter. The **System.Data.SqlTypes.SqlInt64** structure to be converted.

**op\_Implicit**

```
[C#]    public static implicit operator SqlDecimal(SqlMoney x);
[C++]   public: static SqlDecimal op_Implicit(SqlMoney x);
[VB]    returnValue = SqlDecimal.op_Implicit(x)
[JScript]    returnValue = x;
```

*Description*

Converts the **System.Data.SqlTypes.SqlMoney** operand to **System.Data.SqlTypes.SqlDecimal**.

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** equals the **System.Data.SqlTypes.SqlMoney.Value** of the **System.Data.SqlTypes.SqlMoney** parameter. The **System.Data.SqlTypes.SqlMoney** structure to be converted.

**op\_Inequality**

```

1
2 [C#] public static SqlBoolean operator !=(SqlDecimal x, SqlDecimal y);
3 [C++] public: static SqlBoolean op_Inequality(SqlDecimal x, SqlDecimal y);
4 [VB]     returnValue      =      SqlDecimal.op_Inequality(x,      y)
5 [JScript]     returnValue      =      x      !=      y;

```

6

7 *Description*

8 Performs a logical comparison of the two

9 **System.Data.SqlTypes.SqlDecimal** parameters to determine if they are equal.

10 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is

11 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or

12 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either

13 instance of **System.Data.SqlTypes.SqlDecimal** is null, the

14 **System.Data.SqlTypes.SqlBoolean.Value** of the

15 **System.Data.SqlTypes.SqlBoolean** will be

16 **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlDecimal**

17 structure. A **System.Data.SqlTypes.SqlDecimal** structure.

18 op\_LessThan

19

```

20 [C#]     public     static     SqlBoolean     operator
21 [C++] public: static SqlBoolean op_LessThan(SqlDecimal x, SqlDecimal y);
22 [VB]     returnValue      =      SqlDecimal.op_LessThan(x,      y)
23 [JScript]     returnValue      =      x      <      y;

```

24

25 *Description*

1 Performs a logical comparison of two **System.Data.SqlTypes.SqlDecimal**  
2 structures to determine if the first is less than the second.  
3 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
4 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the  
5 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
6 instance of **System.Data.SqlTypes.SqlDecimal** is null, the  
7 **System.Data.SqlTypes.SqlBoolean.Value** of the  
8 **System.Data.SqlTypes.SqlBoolean** will be  
9 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlDecimal**  
10 structure. A **System.Data.SqlTypes.SqlDecimal** structure.

11 op\_LessThanOrEqual

12  
13 [C#] public static SqlBoolean operator <=(SqlDecimal x, SqlDecimal y);

14 [C++] public: static SqlBoolean op\_LessThanOrEqual(SqlDecimal x, SqlDecimal  
15 y);

16 [VB] returnValue = SqlDecimal.op\_LessThanOrEqual(x, y)

17 [JScript] returnValue = x <= y;

## 18 19 *Description*

20 Performs a logical comparison of the two  
21 **System.Data.SqlTypes.SqlDecimal** parameters to determine if the first is less  
22 than or equal to the second.

23 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
24 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal  
25 to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If

either instance of **System.Data.SqlTypes.SqlDecimal** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlDecimal** structure. A **System.Data.SqlTypes.SqlDecimal** structure.

#### op\_Multiply

```
[C#] public static SqlDecimal operator *(SqlDecimal x, SqlDecimal y);
[C++] public: static SqlDecimal op_Multiply(SqlDecimal x, SqlDecimal y);
[VB]     returnValue      =      SqlDecimal.op_Multiply(x,      y)
[JScript]     returnValue      =      x      *      y;
```

#### Description

The multiplication operator computes the product of the two **System.Data.SqlTypes.SqlDecimal** parameters.

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose **System.Data.SqlTypes.SqlDecimal.Value** property contains the product of the multiplication. A **System.Data.SqlTypes.SqlDecimal** structure. A **System.Data.SqlTypes.SqlDecimal** structure.

#### op\_Subtraction

```
[C#] public static SqlDecimal operator -(SqlDecimal x, SqlDecimal y);
[C++] public: static SqlDecimal op_Subtraction(SqlDecimal x, SqlDecimal y);
[VB]     returnValue      =      SqlDecimal.op_Subtraction(x,      y)
[JScript]     returnValue      =      x      -      y;
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

The **System.Data.SqlTypes.subtraction** operator calculates the results of subtracting the second **System.Data.SqlTypes.SqlDecimal** operand from the first.

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose Value property contains the results of the subtraction. A **System.Data.SqlTypes.SqlDecimal** structure. A **System.Data.SqlTypes.SqlDecimal** structure.

**op\_UnaryNegation**

```
[C#]    public    static    SqlDecimal    operator    -(SqlDecimal    x);
[C++]    public:    static    SqlDecimal    op_UnaryNegation(SqlDecimal    x);
[VB]        returnValue    =    SqlDecimal.op_UnaryNegation(x)
[JScript]        returnValue    =    -x;
```

*Description*

The unary minus operator negates the **System.Data.SqlTypes.SqlDecimal** parameter.

*Return Value:* A new **System.Data.SqlTypes.SqlDecimal** structure whose value contains the results of the negation. The **System.Data.SqlTypes.SqlDecimal** structure to be negated.

**Parse**

```
[C#]        public    static    SqlDecimal    Parse(string    s);
[C++]        public:    static    SqlDecimal    Parse(String*    s);
```

1 [VB] Public Shared Function Parse(ByVal s As String) As SqlDecimal  
2 [JScript] public static function Parse(s : String) : SqlDecimal;

3  
4 *Description*

5 [ .][ .]  
6 Power

7  
8 [C#] public static SqlDecimal Power(SqlDecimal n, double exp);  
9 [C++] public: static SqlDecimal Power(SqlDecimal n, double exp);  
10 [VB] Public Shared Function Power(ByVal n As SqlDecimal, ByVal exp As  
11 Double) As SqlDecimal  
12 [JScript] public static function Power(n : SqlDecimal, exp : double) : SqlDecimal;

13  
14 *Description*

15 [ .][ .][ .][ .]  
16 Round

17  
18 [C#] public static SqlDecimal Round(SqlDecimal n, int position);  
19 [C++] public: static SqlDecimal Round(SqlDecimal n, int position);  
20 [VB] Public Shared Function Round(ByVal n As SqlDecimal, ByVal position As  
21 Integer) As SqlDecimal  
22 [JScript] public static function Round(n : SqlDecimal, position : int) : SqlDecimal;

23  
24 *Description*

25 [ .][ .][ .][ .]

1           Sign

2

3   [C#]       public       static       SqlInt32       Sign(SqlDecimal       n);

4   [C++]       public:       static       SqlInt32       Sign(SqlDecimal       n);

5   [VB] Public Shared Function Sign(ByVal n As SqlDecimal) As SqlInt32

6   [JScript] public static function Sign(n : SqlDecimal) : SqlInt32;

7

8   *Description*

9       [ .][ .][ .]

10   Subtract

11

12   [C#] public static SqlDecimal Subtract(SqlDecimal x, SqlDecimal y);

13   [C++] public: static SqlDecimal Subtract(SqlDecimal x, SqlDecimal y);

14   [VB] Public Shared Function Subtract(ByVal x As SqlDecimal, ByVal y As

15       SqlDecimal)                       As                       SqlDecimal

16   [JScript] public static function Subtract(x : SqlDecimal, y : SqlDecimal) :

17       SqlDecimal;

18

19   *Description*

20       [ .]

21   ToDouble

22

23   [C#]                       public                       double                       ToDouble();

24   [C++]                       public:                       double                       ToDouble();

25   [VB]       Public       Function       ToDouble()       As       Double

1 [JScript] public function ToDouble() : double;

2

3 *Description*

4 Returns the a double equal to the contents of the  
5 **System.Data.SqlTypes.SqlDecimal.Value** property of this instance.

6 *Return Value:* The decimal representation of the  
7 **System.Data.SqlTypes.SqlDecimal.Value** property.

8 ToSqlBoolean

9

10 [C#] public SqlBoolean ToSqlBoolean();

11 [C++] public: SqlBoolean ToSqlBoolean();

12 [VB] Public Function ToSqlBoolean() As SqlBoolean

13 [JScript] public function ToSqlBoolean() : SqlBoolean;

14

15 *Description*

16 [ .]

17 ToSqlByte

18

19 [C#] public SqlByte ToSqlByte();

20 [C++] public: SqlByte ToSqlByte();

21 [VB] Public Function ToSqlByte() As SqlByte

22 [JScript] public function ToSqlByte() : SqlByte;

23

24 *Description*

25 [ .]

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

## ToSqlDouble

```
[C#]          public          SqlDouble          ToSqlDouble();
[C++]          public:          SqlDouble          ToSqlDouble();
[VB]      Public      Function ToSqlDouble()      As      SqlDouble
[JScript]      public      function ToSqlDouble()      :      SqlDouble;
```

### Description

[ .]

## ToSqlInt16

```
[C#]          public          SqlInt16          ToSqlInt16();
[C++]          public:          SqlInt16          ToSqlInt16();
[VB]      Public      Function ToSqlInt16()      As      SqlInt16
[JScript]      public      function ToSqlInt16()      :      SqlInt16;
```

### Description

[ .]

## ToSqlInt32

```
[C#]          public          SqlInt32          ToSqlInt32();
[C++]          public:          SqlInt32          ToSqlInt32();
[VB]      Public      Function ToSqlInt32()      As      SqlInt32
[JScript]      public      function ToSqlInt32()      :      SqlInt32;
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

[ .]

ToSqlInt64

[C#]	public	SqlInt64	ToSqlInt64();
[C++]	public:	SqlInt64	ToSqlInt64();
[VB]	Public	Function	ToSqlInt64() As SqlInt64
[JScript]	public	function	ToSqlInt64() : SqlInt64;

*Description*

[ .]

ToSqlMoney

[C#]	public	SqlMoney	ToSqlMoney();
[C++]	public:	SqlMoney	ToSqlMoney();
[VB]	Public	Function	ToSqlMoney() As SqlMoney
[JScript]	public	function	ToSqlMoney() : SqlMoney;

*Description*

[ .]

ToSqlSingle

[C#]	public	SqlSingle	ToSqlSingle();
[C++]	public:	SqlSingle	ToSqlSingle();

1 [VB] Public Function ToSqlSingle() As SqlSingle  
 2 [JScript] public function ToSqlSingle() : SqlSingle;

3  
 4 *Description*

5 [ .]  
 6 ToSqlString

8 [C#] public SqlString ToSqlString();

9 [C++] public: SqlString ToSqlString();

10 [VB] Public Function ToString() As String

11 [JScript] public function ToString() : String;

12  
 13 *Description*

14 [ .]  
 15 ToString

17 [C#] public override string ToString();

18 [C++] public: String\* ToString();

19 [VB] Overrides Public Function ToString() As String

20 [JScript] public override function ToString() : String; Converts a  
 21 **System.Data.SqlTypes.SqlDecimal** structure to **System.String** .

22  
 23 *Description*

24 Converts this **System.Data.SqlTypes.SqlDecimal** structure to  
 25 **System.String** .

1 *Return Value:* A new **System.String** object containing the string representation of  
 2 the **System.Data.SqlTypes.SqlDecimal** structure's  
 3 **System.Data.SqlTypes.SqlDecimal.Value** property.

4 Truncate

6 [C#] public static SqlDecimal Truncate(SqlDecimal n, int position);

7 [C++] public: static SqlDecimal Truncate(SqlDecimal n, int position);

8 [VB] Public Shared Function Truncate(ByVal n As SqlDecimal, ByVal position

9 As Integer) As SqlDecimal

10 [JScript] public static function Truncate(n : SqlDecimal, position : int) :

11 SqlDecimal;

13 *Description*

14 [ .][ .][ .][ .]

15 SqlDouble structure (System.Data.SqlTypes)

16 Truncate

19 *Description*

20 Represents a floating-point number within the range of -1.79E +308  
 21 through 1.79E +308 to be stored in or retrieved from a database.

22 Truncate

24 [C#] public static readonly SqlDouble MaxValue;

25 [C++] public: static SqlDouble MaxValue;

1 [VB] Public Shared ReadOnly MaxValue As SqlDouble  
2 [JScript] public static var MaxValue : SqlDouble;

3  
4 *Description*

5 A constant representing the maximum value for a  
6 **System.Data.SqlTypes.SqlDouble** structure.

7 This value is 1.79E+308 [ .]

8 Truncate

9  
10 [C#] public static readonly SqlDouble MinValue;

11 [C++] public: static SqlDouble MinValue;

12 [VB] Public Shared ReadOnly MinValue As SqlDouble

13 [JScript] public static var MinValue : SqlDouble;

14  
15 *Description*

16 A constant representing the minimum possible value of  
17 **System.Data.SqlTypes.SqlDouble** .

18 This value is -1.79E+308 [ .]

19 Truncate

20  
21 [C#] public static readonly SqlDouble Null;

22 [C++] public: static SqlDouble Null;

23 [VB] Public Shared ReadOnly Null As SqlDouble

24 [JScript] public static var Null : SqlDouble;

25

*Description*

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlDouble.Value** property of an instance of the **System.Data.SqlTypes.SqlDouble** structure.

**System.Data.SqlTypes.SqlDouble.Null** functions as a constant for the **System.Data.SqlTypes.SqlDouble** structure.

Truncate

[C#]	public	static	readonly	SqlDouble	Zero;
[C++]	public:	static		SqlDouble	Zero;
[VB]	Public	Shared	ReadOnly	Zero	As SqlDouble
[JScript]	public	static	var	Zero	: SqlDouble;

*Description*

Represents a zero value that can be assigned to the **System.Data.SqlTypes.SqlDouble.Value** property of an instance of the **System.Data.SqlTypes.SqlDouble** structure.

The **System.Data.SqlTypes.SqlDouble.Zero** field is a constant for the **System.Data.SqlTypes.SqlDouble** structure.

SqlDouble

*Example Syntax:*

Truncate

[C#]	public	SqlDouble(double	value);
------	--------	------------------	---------

```

1  [C++]          public:          SqlDouble(double          value);
2  [VB]    Public    Sub    New(ByVal    value    As    Double)
3  [JScript]    public    function    SqlDouble(value    :    double);

```

4  
5 *Description*

6        Initializes a new instance of the **System.Data.SqlTypes.SqlDouble**  
7 structure using the supplied double parameter to set the new SqlDouble structure's  
8 **System.Data.SqlTypes.SqlDouble.Value** property. A double whose value will be  
9 used for the new **System.Data.SqlTypes.SqlDouble**.

10        IsNull

11        Truncate

```

12
13 [C#]          public          bool          IsNull          {get;}
14 [C++]          public:          __property          bool          get_IsNull();
15 [VB]    Public    ReadOnly    Property    IsNull    As    Boolean
16 [JScript]    public    function    get    IsNull()    :    Boolean;

```

17  
18 *Description*

19        Indicates whether or not **System.Data.SqlTypes.SqlDouble.Value** is null.

20        Value

21        Truncate

```

22
23 [C#]          public          double          Value          {get;}
24 [C++]          public:          __property          double          get_Value();
25 [VB]    Public    ReadOnly    Property    Value    As    Double

```

1 [JScript] public function get Value() : double;

2

3 *Description*

4 Gets the value of the **System.Data.SqlTypes.SqlDouble** structure. This  
5 property is read-only.

6 Add

7

8 [C#] public static SqlDouble Add(SqlDouble x, SqlDouble y);

9 [C++] public: static SqlDouble Add(SqlDouble x, SqlDouble y);

10 [VB] Public Shared Function Add(ByVal x As SqlDouble, ByVal y As

11 SqlDouble) As SqlDouble

12 [JScript] public static function Add(x : SqlDouble, y : SqlDouble) : SqlDouble;

13

14 *Description*

15 [ .]

16 CompareTo

17

18 [C#] public int CompareTo(object value);

19 [C++] public: \_\_sealed int CompareTo(Object\* value);

20 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

21 Integer

22 [JScript] public function CompareTo(value : Object) : int;

23

24 *Description*

25

Compares this instance to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to compare.

#### Divide

[C#] public static SqlDouble Divide(SqlDouble x, SqlDouble y);

[C++] public: static SqlDouble Divide(SqlDouble x, SqlDouble y);

[VB] Public Shared Function Divide(ByVal x As SqlDouble, ByVal y As SqlDouble) As SqlDouble

[JScript] public static function Divide(x : SqlDouble, y : SqlDouble) : SqlDouble;

#### Description

[ . ]

#### Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object\* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean;

#### Description

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlDateTime.Value** property of the **System.Data.SqlTypes.SqlDouble** object.

*Return Value:* **true** if object is an instance of **System.Data.SqlTypes.SqlByte** and the two are equal; otherwise **false** . The object to be compared.

### Equals

```
[C#] public static new SqlBoolean Equals(SqlDouble x, SqlDouble y);
[C++] public: static SqlBoolean Equals(SqlDouble x, SqlDouble y);
[VB] Shadows Public Shared Function Equals(ByVal x As SqlDouble, ByVal y
As SqlDouble) As SqlBoolean
[JavaScript] public static hide function Equals(x : SqlDouble, y : SqlDouble) :
SqlBoolean;
```

### Description

[ .]

### GetHashCode

```
[C#] public override int GetHashCode();
[C++] public: int GetHashCode();
[VB] Overrides Public Function GetHashCode() As Integer
[JavaScript] public override function GetHashCode() : int;
```

### Description

Returns the hash code for this instance.

*Return Value:* A 32-bit signed integer hash code.

### GreaterThan

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public static SqlBoolean GreaterThan(SqlDouble x, SqlDouble y);
[C++] public: static SqlBoolean GreaterThan(SqlDouble x, SqlDouble y);
[VB] Public Shared Function GreaterThan(ByVal x As SqlDouble, ByVal y As
SqlDouble)
As
SqlBoolean
[JScript] public static function GreaterThan(x : SqlDouble, y : SqlDouble) :
SqlBoolean;
```

*Description*

[ .]  
GreaterThanOrEqual

```
[C#] public static SqlBoolean GreaterThanOrEqual(SqlDouble x, SqlDouble y);
[C++] public: static SqlBoolean GreaterThanOrEqual(SqlDouble x, SqlDouble y);
[VB] Public Shared Function GreaterThanOrEqual(ByVal x As SqlDouble, ByVal
y
As
SqlDouble)
As
SqlBoolean
[JScript] public static function GreaterThanOrEqual(x : SqlDouble, y : SqlDouble)
:
SqlBoolean;
```

*Description*

[ .]  
LessThan

```
[C#] public static SqlBoolean LessThan(SqlDouble x, SqlDouble y);
[C++] public: static SqlBoolean LessThan(SqlDouble x, SqlDouble y);
```

1 [VB] Public Shared Function LessThan(ByVal x As SqlDouble, ByVal y As  
2 SqlDouble) As SqlBoolean  
3 [JScript] public static function LessThan(x : SqlDouble, y : SqlDouble) :  
4 SqlBoolean;

5  
6 *Description*

7 [ .]  
8 LessThanOrEqual

9  
10 [C#] public static SqlBoolean LessThanOrEqual(SqlDouble x, SqlDouble y);  
11 [C++] public: static SqlBoolean LessThanOrEqual(SqlDouble x, SqlDouble y);  
12 [VB] Public Shared Function LessThanOrEqual(ByVal x As SqlDouble, ByVal y  
13 As SqlDouble) As SqlBoolean  
14 [JScript] public static function LessThanOrEqual(x : SqlDouble, y : SqlDouble) :  
15 SqlBoolean;

16  
17 *Description*

18 [ .]  
19 Multiply

20  
21 [C#] public static SqlDouble Multiply(SqlDouble x, SqlDouble y);  
22 [C++] public: static SqlDouble Multiply(SqlDouble x, SqlDouble y);  
23 [VB] Public Shared Function Multiply(ByVal x As SqlDouble, ByVal y As  
24 SqlDouble) As SqlDouble  
25 [JScript] public static function Multiply(x : SqlDouble, y : SqlDouble) :

SqlDouble;

### Description

$$[.]$$

## NotEquals

```
[C#] public static SqlBoolean NotEquals(SqlDouble x, SqlDouble y);
```

```
[C++] public: static SqlBoolean NotEquals(SqlDouble x, SqlDouble y);
```

[VB] Public Shared Function NotEquals(ByVal x As SqlDouble, ByVal y As

SqlDouble)	As	SqlBoolean
------------	----	------------

```
[JScript] public static function NotEquals(x : SqlDouble, y : SqlDouble) :
```

SqlBoolean;

### Description

$$[.]$$

op\_Addition

```
[C#] public static SqlDouble operator +(SqlDouble x, SqlDouble y);
```

```
[C++] public: static SqlDouble op_Addition(SqlDouble x, SqlDouble y);
```

```
[VB]      returnValue      =      SqlDouble.op_Addition(x,      y)
```

```
[JScript]      returnValue      =      x      +      y;
```

### Description

The addition operator computes the sum of the two

**System.Data.SqlTypes.SqlDouble** operands.

1 *Return Value:* The sum of the two **System.Data.SqlTypes.SqlDouble** operands.

2 A **System.Data.SqlTypes.SqlDouble** structure. A

3 **System.Data.SqlTypes.SqlDouble** structure.

4 op\_Division

6 [C#] public static SqlDouble operator /(SqlDouble x, SqlDouble y);

7 [C++] public: static SqlDouble op\_Division(SqlDouble x, SqlDouble y);

8 [VB] returnValue = SqlDouble.op\_Division(x, y)

9 [JScript] returnValue = x / y;

11 *Description*

12 The division operator divides the first **System.Data.SqlTypes.SqlDouble**  
13 operand by the second.

14 *Return Value:* The results of the division operation. A

15 **System.Data.SqlTypes.SqlDouble** structure. A

16 **System.Data.SqlTypes.SqlDouble** structure.

17 op\_Equality

19 [C#] public static SqlBoolean operator ==(SqlDouble x, SqlDouble y);

20 [C++] public: static SqlBoolean op\_Equality(SqlDouble x, SqlDouble y);

21 [VB] returnValue = SqlDouble.op\_Equality(x, y)

22 [JScript] returnValue = x == y;

24 *Description*

25

Performs a logical comparison on two instances of **System.Data.SqlTypes.SqlDouble** to determine if they are equal.

*Return Value:* **true** if the two values are equal, otherwise **false**. A **System.Data.SqlTypes.SqlDouble** structure. A **System.Data.SqlTypes.SqlDouble** structure.

op\_Explicit

[C#] public static explicit operator SqlDouble(SqlBoolean x);

[C++] public: static SqlDouble op\_Explicit(SqlBoolean x);

[VB] returnValue = SqlDouble.op\_Explicit(x)

[JScript] returnValue = SqlDouble(x);

### Description

Converts the supplied **System.Data.SqlTypes.SqlBit** parameter to **System.Data.SqlTypes.SqlDouble**.

*Return Value:* A new **System.Data.SqlTypes.SqlDouble** structure whose **System.Data.SqlTypes.SqlDouble.Value** is either 0 or 1, depending on the **System.Data.SqlTypes.SqlBit.ByteValue** property of the **System.Data.SqlTypes.SqlBit** parameter. The **System.Data.SqlTypes.SqlBit** to be converted.

op\_Explicit

[C#] public static explicit operator double(SqlDouble x);

[C++] public: static double op\_Explicit();

[VB] returnValue = SqlDouble.op\_Explicit(x)

1 [JScript]                   returnValue                   =                   Double(x);

2

3 *Description*

4       Converts the supplied **System.Data.SqlTypes.SqlDouble** structure to  
5 double. A **System.Data.SqlTypes.SqlDouble** structure.

6       op\_Explicit

7

8 [C#]     public     static     explicit     operator     SqlDouble(SqlString   x);

9 [C++]    public:    static     SqlDouble     op\_Explicit(SqlString   x);

10 [VB]       returnValue                   =               SqlDouble.op\_Explicit(x)

11 [JScript]       returnValue                   =               SqlDouble(x);

12

13 *Description*

14       Converts the supplied **System.Data.SqlTypes.SqlString** parameter to  
15 **System.Data.SqlTypes.SqlDouble**

16 *Return Value:*   A new **System.Data.SqlTypes.SqlDouble** whose  
17 **System.Data.SqlTypes.SqlDouble.Value** is equal to the value of the number  
18 represented by the **System.Data.SqlTypes.SqlString** . A **SqlString** object.

19       op\_GreaterThan

20

21 [C#]     public     static     SqlBoolean     operator   >(SqlDouble x, SqlDouble y);

22 [C++]    public:    static     SqlBoolean     op\_GreaterThan(SqlDouble x, SqlDouble y);

23 [VB]       returnValue                   =               SqlDouble.op\_GreaterThan(x,       y)

24 [JScript]       returnValue                   =               x               >               y;

25

## Description

Compares two instances of **System.Data.SqlTypes.SqlDouble** to determine if the first is greater than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlDouble** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlDouble** structure. A **System.Data.SqlTypes.SqlDouble** structure.

op\_GreaterThanOrEqual

[C#] public static SqlBoolean operator >=(SqlDouble x, SqlDouble y);

[C++] public: static SqlBoolean op\_GreaterThanOrEqual(SqlDouble x, SqlDouble y);

[VB] returnValue = SqlDouble.op\_GreaterThanOrEqual(x, y)

[JScript] returnValue = x >= y;

## Description

Compares two instances of **System.Data.SqlTypes.SqlDouble** to determine if the first is greater than or equal to the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or

1 equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**  
 2 . If either instance of **System.Data.SqlTypes.SqlDouble** is null, the  
 3 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 4 **System.Data.SqlTypes.SqlBoolean** will be  
 5 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlDouble**  
 6 structure. A **System.Data.SqlTypes.SqlDouble** structure.

7 **op\_Implicit**

8  
 9 [C#] public static implicit operator SqlDouble(double x);  
 10 [C++] public: static SqlDouble op\_Implicit(double x);  
 11 [VB] returnValue = SqlDouble.op\_Implicit(x)  
 12 [JScript] returnValue = x;

13  
 14 *Description*

15 Converts the supplied double value to a **System.Data.SqlTypes.SqlDouble**  
 16 . The double value to convert.

17 **op\_Implicit**

18  
 19 [C#] public static implicit operator SqlDouble(SqlByte x);  
 20 [C++] public: static SqlDouble op\_Implicit(SqlByte x);  
 21 [VB] returnValue = SqlDouble.op\_Implicit(x)  
 22 [JScript] returnValue = x;

23  
 24 *Description*

25

Converts the supplied **System.Data.SqlTypes.SqlByte** parameter to **System.Data.SqlTypes.SqlDouble**.

*Return Value:* A **System.Data.SqlTypes.SqlDouble** structure whose **System.Data.SqlTypes.SqlDouble.Value** is equal to the **System.Data.SqlTypes.SqlByte.Value** of the **System.Data.SqlTypes.SqlByte** parameter. A **System.Data.SqlTypes.SqlDouble** structure.

op\_Explicit

```
[C#]    public static implicit operator SqlDouble(SqlDecimal x);
[C++]    public: static SqlDouble op_Explicit(SqlDecimal x);
[VB]        returnValue = SqlDouble.op_Explicit(x)
[JScript]        returnValue = x;
```

#### Description

Converts the supplied **System.Data.SqlTypes.SqlDecimal** parameter to **System.Data.SqlTypes.SqlDouble**.

*Return Value:* A new **System.Data.SqlTypes.SqlDouble** structure whose **System.Data.SqlTypes.SqlDouble.Value** is equal to the **System.Data.SqlTypes.SqlDecimal.Value** of the **System.Data.SqlTypes.SqlDecimal** parameter. A **System.Data.SqlTypes.SqlDecimal** structure.

op\_Explicit

```
[C#]    public static implicit operator SqlDouble(SqlInt16 x);
[C++]    public: static SqlDouble op_Explicit(SqlInt16 x);
```

```

1  [VB]          returnValue          =          SqlDouble.op_Implicit(x)
2  [JScript]          returnValue          =          x;

```

3

4 *Description*

5       Converts the supplied **System.Data.SqlTypes.SqlInt16** parameter to

6 **System.Data.SqlTypes.SqlDouble** .

7 *Return Value:* A new **System.Data.SqlTypes.SqlDouble** structure whose

8 **System.Data.SqlTypes.SqlDouble.Value** is equal to the

9 **System.Data.SqlTypes.SqlInt16.Value** of the **System.Data.SqlTypes.SqlInt16**

10 parameter. A **System.Data.SqlTypes.SqlInt16** structure.

11       op\_Implicit

```

12
13 [C#]   public   static   implicit   operator   SqlDouble(SqlInt32   x);
14 [C++]   public:   static   SqlDouble   op_Implicit(SqlInt32   x);
15 [VB]          returnValue          =          SqlDouble.op_Implicit(x)
16 [JScript]          returnValue          =          x;

```

17

18 *Description*

19       Converts the supplied **System.Data.SqlTypes.SqlInt32** parameter to

20 **System.Data.SqlTypes.SqlDouble** .

21 *Return Value:* A new **System.Data.SqlTypes.SqlDouble** whose

22 **System.Data.SqlTypes.SqlDouble.Value** is equal to the

23 **System.Data.SqlTypes.SqlInt32.Value** of the **System.Data.SqlTypes.SqlInt32**

24 parameter. A **System.Data.SqlTypes.SqlInt32** structure.

25       op\_Implicit

```

1
2 [C#]    public    static    implicit    operator    SqlDouble(SqlInt64    x);
3 [C++]   public:   static    SqlDouble    op_Implicit(SqlInt64    x);
4 [VB]    returnValue    =    SqlDouble.op_Implicit(x)
5 [JScript]    returnValue    =    x;

```

6  
7 *Description*

8 Converts the supplied **System.Data.SqlTypes.SqlInt64** parameter to  
9 **System.Data.SqlTypes.SqlDouble**  
10 *Return Value:* A new **System.Data.SqlTypes.SqlDouble** whose  
11 **System.Data.SqlTypes.SqlDouble.Value** is equal to the  
12 **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64**  
13 parameter. A **System.Data.SqlTypes.SqlInt64** structure.

14 op\_Implicit

```

15
16 [C#]    public    static    implicit    operator    SqlDouble(SqlMoney    x);
17 [C++]   public:   static    SqlDouble    op_Implicit(SqlMoney    x);
18 [VB]    returnValue    =    SqlDouble.op_Implicit(x)
19 [JScript]    returnValue    =    x;

```

20  
21 *Description*

22 Converts the supplied **System.Data.SqlTypes.SqlMoney** parameter to  
23 **System.Data.SqlTypes.SqlDouble**  
24 *Return Value:* A new **System.Data.SqlTypes.SqlDouble** whose  
25 **System.Data.SqlTypes.SqlDouble.Value** is equal to the

1 **System.Data.SqlTypes.SqlMoney.Value** of the  
 2 **System.Data.SqlTypes.SqlMoney** parameter. A  
 3 **System.Data.SqlTypes.SqlMoney** structure.

4 **op\_Implicit**

5  
 6 [C#] public static implicit operator SqlDouble(SqlSingle x);  
 7 [C++] public: static SqlDouble op\_Implicit(SqlSingle x);  
 8 [VB] returnValue = SqlDouble.op\_Implicit(x)  
 9 [JScript] returnValue = x;

10  
 11 *Description*

12 Converts the supplied **System.Data.SqlTypes.SqlSingle** parameter to  
 13 **System.Data.SqlTypes.SqlDouble**.  
 14 *Return Value:* A new **System.Data.SqlTypes.SqlDouble** structure whose  
 15 **System.Data.SqlTypes.SqlDouble.Value** is equal to the  
 16 **System.Data.SqlTypes.SqlSingle.Value** of the **System.Data.SqlTypes.SqlSingle**  
 17 parameter. A **System.Data.SqlTypes.SqlSingle** structure.

18 **op\_Inequality**

19  
 20 [C#] public static SqlBoolean operator !=(SqlDouble x, SqlDouble y);  
 21 [C++] public: static SqlBoolean op\_Inequality(SqlDouble x, SqlDouble y);  
 22 [VB] returnValue = SqlDouble.op\_Inequality(x, y)  
 23 [JScript] returnValue = x != y;

24  
 25 *Description*

1 Compares two instances of **System.Data.SqlTypes.SqlDouble** to  
2 determine if they are equal.

3 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
4 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or  
5 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either  
6 instance of **System.Data.SqlTypes.SqlDouble** is null, the  
7 **System.Data.SqlTypes.SqlBoolean.Value** of the  
8 **System.Data.SqlTypes.SqlBoolean** will be  
9 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlDouble**  
10 structure. A **System.Data.SqlTypes.SqlDouble** structure.

11 op\_LessThan

12  
13 [C#] public static SqlBoolean operator

14 [C++] public: static SqlBoolean op\_LessThan(SqlDouble x, SqlDouble y);

15 [VB] returnValue = SqlDouble.op\_LessThan(x, y)

16 [JScript] returnValue = x < y;

17  
18 *Description*

19 Compares two instances of **System.Data.SqlTypes.SqlDouble** to  
20 determine if the first is less than the second.

21 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
22 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the  
23 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
24 instance of **System.Data.SqlTypes.SqlDouble** is null, the  
25 **System.Data.SqlTypes.SqlBoolean.Value** of the

1 **System.Data.SqlTypes.SqlBoolean** will be  
 2 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlDouble**  
 3 structure. A **System.Data.SqlTypes.SqlDouble** structure.

4 **op\_LessThanOrEqual**

5  
 6 [C#] public static SqlBoolean operator <=(SqlDouble x, SqlDouble y);  
 7 [C++] public: static SqlBoolean op\_LessThanOrEqual(SqlDouble x, SqlDouble  
 8 y);  
 9 [VB] returnValue = SqlDouble.op\_LessThanOrEqual(x, y)  
 10 [JScript] returnValue = x <= y;

11  
 12 *Description*

13 Compares two instances of **System.Data.SqlTypes.SqlDouble** to  
 14 determine if the first is less than or equal to the second.

15 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 16 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal  
 17 to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If  
 18 either instance of **System.Data.SqlTypes.SqlDouble** is null, the  
 19 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 20 **System.Data.SqlTypes.SqlBoolean** will be  
 21 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlDouble**  
 22 structure. A **System.Data.SqlTypes.SqlDouble** structure.

23 **op\_Multiply**

24  
 25 [C#] public static SqlDouble operator \*(SqlDouble x, SqlDouble y);

1 [C++] public: static SqlDouble op\_Multiply(SqlDouble x, SqlDouble y);  
 2 [VB]       returnValue       =       SqlDouble.op\_Multiply(x,       y)  
 3 [JScript]       returnValue       =       x       \*       y;

4  
 5 *Description*

6       The multiplication operator computes the product of the two  
 7 **System.Data.SqlTypes.SqlDouble** operands.

8 *Return Value:* The product of the two **System.Data.SqlTypes.SqlDouble**  
 9 operands. A **System.Data.SqlTypes.SqlDouble** structure. A  
 10 **System.Data.SqlTypes.SqlDouble** structure.

11       op\_Subtraction

12  
 13 [C#] public static SqlDouble operator -(SqlDouble x, SqlDouble y);  
 14 [C++] public: static SqlDouble op\_Subtraction(SqlDouble x, SqlDouble y);  
 15 [VB]       returnValue       =       SqlDouble.op\_Subtraction(x,       y)  
 16 [JScript]       returnValue       =       x       -       y;

17  
 18 *Description*

19       The subtraction operator subtracts the second **System.Data.SqlTypes.SqlDouble**  
 20 operand from the first.

21 *Return Value:* The results of the subtraction operation. A  
 22 **System.Data.SqlTypes.SqlDouble** structure. A  
 23 **System.Data.SqlTypes.SqlDouble** structure.

24       op\_UnaryNegation

25

```

1
2 [C#]      public      static      SqlDouble      operator      -(SqlDouble      x);
3 [C++]     public:     static      SqlDouble      op_UnaryNegation(SqlDouble      x);
4 [VB]      returnValue      =      SqlDouble.op_UnaryNegation(x)
5 [JScript]      returnValue      =      -x;

```

6

7 *Description*

8 Returns the negated value of the **System.Data.SqlTypes.SqlDouble**

9 operand. A **System.Data.SqlTypes.SqlDouble** structure.

10 Parse

11

```

12 [C#]      public      static      SqlDouble      Parse(string      s);
13 [C++]     public:     static      SqlDouble      Parse(String*      s);
14 [VB]      Public Shared Function Parse(ByVal s As String) As SqlDouble
15 [JScript] public static function Parse(s : String) : SqlDouble;

```

16

17 *Description*

18 [ . ][ . ]

19 Subtract

20

```

21 [C#]      public      static      SqlDouble      Subtract(SqlDouble      x, SqlDouble      y);
22 [C++]     public:     static      SqlDouble      Subtract(SqlDouble      x, SqlDouble      y);
23 [VB]      Public Shared Function Subtract(ByVal x As SqlDouble, ByVal y As
24 SqlDouble)      As      SqlDouble
25 [JScript] public static function Subtract(x : SqlDouble, y : SqlDouble) :

```

```

1  SqlDouble;
2
3  Description
4      [ .]
5      ToSqlBoolean
6
7  [C#]          public          SqlBoolean          ToSqlBoolean();
8  [C++]         public:         SqlBoolean          ToSqlBoolean();
9  [VB]          Public          Function            ToSqlBoolean()  As      SqlBoolean
10 [JScript]      public          function            ToSqlBoolean()  :      SqlBoolean;

```

```

11
12 Description
13     [ .]
14     ToSqlByte
15
16 [C#]          public          SqlByte          ToSqlByte();
17 [C++]         public:         SqlByte          ToSqlByte();
18 [VB]          Public          Function            ToSqlByte()      As      SqlByte
19 [JScript]      public          function            ToSqlByte()      :      SqlByte;

```

```

20
21 Description
22     [ .]
23     ToSqlDecimal
24
25 [C#]          public          SqlDecimal         ToSqlDecimal();

```

```

1  [C++]          public:          SqlDecimal          ToSqlDecimal();
2  [VB]      Public      Function  ToSqlDecimal()      As      SqlDecimal
3  [JScript]   public      function  ToSqlDecimal()      :      SqlDecimal;

```

4

5 *Description*

6 [ .]

7 ToSqlInt16

8

```

9  [C#]          public          SqlInt16          ToSqlInt16();
10 [C++]          public:          SqlInt16          ToSqlInt16();
11 [VB]      Public      Function  ToSqlInt16()      As      SqlInt16
12 [JScript]   public      function  ToSqlInt16()      :      SqlInt16;

```

13

14 *Description*

15 [ .]

16 ToSqlInt32

17

```

18 [C#]          public          SqlInt32          ToSqlInt32();
19 [C++]          public:          SqlInt32          ToSqlInt32();
20 [VB]      Public      Function  ToSqlInt32()      As      SqlInt32
21 [JScript]   public      function  ToSqlInt32()      :      SqlInt32;

```

22

23 *Description*

24 [ .]

25 ToSqlInt64

```

1
2 [C#]          public          SqlInt64          ToSqlInt64();
3 [C++]         public:         SqlInt64          ToSqlInt64();
4 [VB]         Public          Function          ToSqlInt64() As          SqlInt64
5 [JScript]     public          function          ToSqlInt64() :          SqlInt64;
6

```

7 *Description*

```

8     [ .]
9     ToSqlMoney
10

```

```

11 [C#]          public          SqlMoney          ToSqlMoney();
12 [C++]         public:         SqlMoney          ToSqlMoney();
13 [VB]         Public          Function          ToSqlMoney() As          SqlMoney
14 [JScript]     public          function          ToSqlMoney() :          SqlMoney;
15

```

16 *Description*

```

17     [ .]
18     ToSqlSingle
19

```

```

20 [C#]          public          SqlSingle          ToSqlSingle();
21 [C++]         public:         SqlSingle          ToSqlSingle();
22 [VB]         Public          Function          ToSqlSingle() As          SqlSingle
23 [JScript]     public          function          ToSqlSingle() :          SqlSingle;
24

```

25 *Description*



U  
S  
E  
S  
O  
F  
T  
W  
A  
R  
E

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

1 [JScript] public function SqlGuid(value : Byte[]); Initializes a new instance of the  
 2 **System.Data.SqlTypes.SqlGuid** structure.

3  
 4 *Description*

5 Initializes a new instance of the **System.Data.SqlTypes.SqlGuid** structure  
 6 using the supplied byte array parameter. A byte array.

7 SqlGuid

8 *Example Syntax:*

9 ToString

10  
 11 [C#] public SqlGuid(Guid g);

12 [C++] public: SqlGuid(Guid g);

13 [VB] Public Sub New(ByVal g As Guid)

14 [JScript] public function SqlGuid(g : Guid);

15  
 16 *Description*

17 Initializes a new instance of the **System.Data.SqlTypes.SqlGuid** structure  
 18 using the supplied **System.Guid** parameter. A **System.Guid**

19 SqlGuid

20 *Example Syntax:*

21 ToString

22  
 23 [C#] public SqlGuid(string s);

24 [C++] public: SqlGuid(String\* s);

25 [VB] Public Sub New(ByVal s As String)



```

1
2 [C#]          public          bool          IsNull          {get;}
3 [C++]         public:         __property    bool          get_IsNull();
4 [VB]   Public   ReadOnly   Property   IsNull   As   Boolean
5 [JScript]     public   function   get   IsNull()   :   Boolean;

```

```

6
7 Description
8     Indicates whether or not System.Data.SqlTypes.SqlGuid.Value is null.
9     Value
10    ToString

```

```

11
12 [C#]          public          Guid          Value          {get;}
13 [C++]         public:         __property    Guid          get_Value();
14 [VB]   Public   ReadOnly   Property   Value   As   Guid
15 [JScript]     public   function   get   Value()   :   Guid;

```

```

16
17 Description
18     Gets the value of the System.Data.SqlTypes.SqlGuid structure. This
19 property is read-only.
20     CompareTo

```

```

21
22 [C#]          public          int          CompareTo(object          value);
23 [C++]         public:         __sealed    int          CompareTo(Object*          value);
24 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
25 Integer

```

[JScript] public function CompareTo(value : Object) : int;

### Description

Compares this **System.Data.SqlTypes.SqlGuid** structure to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

### Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object\* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean;

### Description

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlGuid.Value** property of the **System.Data.SqlTypes.SqlGuid** object.

*Return Value:* **true** if object is an instance of **SqlGuid** and the two are equal; otherwise **false** . The object to be compared.

### Equals

[C#] public static new SqlBoolean Equals(SqlGuid x, SqlGuid y);

[C++] public: static SqlBoolean Equals(SqlGuid x, SqlGuid y);

[VB] Shadows Public Shared Function Equals(ByVal x As SqlGuid, ByVal y As



1 [ .]  
2 GreaterThanOrEqualTo  
3  
4 [C#] public static SqlBoolean GreaterThanOrEqualTo(SqlGuid x, SqlGuid y);  
5 [C++] public: static SqlBoolean GreaterThanOrEqualTo(SqlGuid x, SqlGuid y);  
6 [VB] Public Shared Function GreaterThanOrEqualTo(ByVal x As SqlGuid, ByVal y  
7 As SqlGuid) As SqlBoolean  
8 [JScript] public static function GreaterThanOrEqualTo(x : SqlGuid, y : SqlGuid) :  
9 SqlBoolean;

10  
11 *Description*

12 [ .]  
13 LessThan  
14  
15 [C#] public static SqlBoolean LessThan(SqlGuid x, SqlGuid y);  
16 [C++] public: static SqlBoolean LessThan(SqlGuid x, SqlGuid y);  
17 [VB] Public Shared Function LessThan(ByVal x As SqlGuid, ByVal y As  
18 SqlGuid) As SqlBoolean  
19 [JScript] public static function LessThan(x : SqlGuid, y : SqlGuid) : SqlBoolean;

20  
21 *Description*

22 [ .]  
23 LessThanOrEqualTo  
24  
25 [C#] public static SqlBoolean LessThanOrEqualTo(SqlGuid x, SqlGuid y);

```

1  [C++] public: static SqlBoolean LessThanOrEqual(SqlGuid x, SqlGuid y);
2  [VB] Public Shared Function LessThanOrEqual(ByVal x As SqlGuid, ByVal y As
3  SqlGuid)                                As                                SqlBoolean
4  [JScript] public static function LessThanOrEqual(x : SqlGuid, y : SqlGuid) :
5  SqlBoolean;

```

```

6
7  Description

```

```

8      [ .]
9      NotEquals

```

```

11 [C#]  public static SqlBoolean NotEquals(SqlGuid x, SqlGuid y);
12 [C++] public: static SqlBoolean NotEquals(SqlGuid x, SqlGuid y);
13 [VB]  Public Shared Function NotEquals(ByVal x As SqlGuid, ByVal y As
14 SqlGuid)                                As                                SqlBoolean
15 [JScript] public static function NotEquals(x : SqlGuid, y : SqlGuid) : SqlBoolean;

```

```

16
17 Description

```

```

18      [ .]
19      op_Equality

```

```

21 [C#]  public static SqlBoolean operator ==(SqlGuid x, SqlGuid y);
22 [C++] public: static SqlBoolean op_Equality(SqlGuid x, SqlGuid y);
23 [VB]      returnValue      =      SqlGuid.op_Equality(x,      y)
24 [JScript]      returnValue      =      x      ==      y;

```

```

25

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Performs a logical comparison of two **System.Data.SqlTypes.SqlGuid** structures to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If either instance of **SqlGuid** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlGuid** structure. A **SqlGuid** structure.

op\_Explicit

```
[C#]    public static explicit operator SqlGuid(SqlBinary x);
[C++]    public: static SqlGuid op_Explicit(SqlBinary x);
[VB]    returnValue = SqlGuid.op_Explicit(x)
[JScript]    returnValue = SqlGuid(x);
```

*Description*

Converts the **System.Data.SqlTypes.SqlBinary** parameter to **System.Data.SqlTypes.SqlGuid**

*Return Value:* A new **SqlGuid** whose **System.Data.SqlTypes.SqlGuid.Value** is equal to the **System.Data.SqlTypes.SqlBinary.Value** of the **SqlBinary** parameter. A **SqlBinary** object.

op\_Explicit

```

1
2 [C#]      public      static      explicit      operator      Guid(SqlGuid      x);
3 [C++]      public:      static      Guid      op_Explicit();
4 [VB]      returnValue      =      SqlGuid.op_Explicit(x)
5 [JScript]      returnValue      =      Guid(x);

```

```

6
7 Description
8
9     Converts the supplied System.Data.SqlTypes.SqlGuid parameter to
10 System.Guid
11 Return Value: A new Guid equal to the System.Data.SqlTypes.SqlGuid.Value
12 of the SqlGuid . A SqlGuid structure.

```

```

13
14 [C#]      public      static      explicit      operator      SqlGuid(SqlString      x);
15 [C++]      public:      static      SqlGuid      op_Explicit(SqlString      x);
16 [VB]      returnValue      =      SqlGuid.op_Explicit(x)
17 [JScript]      returnValue      =      SqlGuid(x);

```

```

18
19 Description
20
21     Converts the supplied System.Data.SqlTypes.SqlString object parameter
22 to System.Data.SqlTypes.SqlGuid
23 Return Value: A SqlGuid whose System.Data.SqlTypes.SqlGuid.Value equals
24 the value represented by the String parameter. A SqlString object.

```

```

25
26     op_GreaterThan

```

```

1
2 [C#] public static SqlBoolean operator >(SqlGuid x, SqlGuid y);
3 [C++] public: static SqlBoolean op_GreaterThan(SqlGuid x, SqlGuid y);
4 [VB]     returnValue      =      SqlGuid.op_GreaterThan(x,      y)
5 [JScript]     returnValue      =      x      >      y;

```

### Description

Compares two instances of **System.Data.SqlTypes.SqlGuid** to determine if the first is greater than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **SqlGuid** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlGuid** structure. A **SqlGuid** structure.

op\_GreaterThanOrEqual

```

17
18 [C#] public static SqlBoolean operator >=(SqlGuid x, SqlGuid y);
19 [C++] public: static SqlBoolean op_GreaterThanOrEqual(SqlGuid x, SqlGuid y);
20 [VB]     returnValue      =      SqlGuid.op_GreaterThanOrEqual(x,      y)
21 [JScript]     returnValue      =      x      >=      y;

```

### Description

Compares two instances of **System.Data.SqlTypes.SqlGuid** to determine if the first is greater than or equal to the second.

1 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 2 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greaater than or  
 3 equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**  
 4 . If either instance of **SqlGuid** is null, the  
 5 **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be  
 6 **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlGuid** structure. A **SqlGuid**  
 7 structure.

8 **op\_Implicit**

9  
 10 [C#] public static implicit operator SqlGuid(Guid x);  
 11 [C++] public: static SqlGuid op\_Implicit(Guid x);  
 12 [VB] returnValue = SqlGuid.op\_Implicit(x)  
 13 [JScript] returnValue = x;

14  
 15 *Description*

16 Converts the supplied **System.Guid** parameter to  
 17 **System.Data.SqlTypes.SqlGuid**

18 *Return Value:* A new **SqlGuid** whose **System.Data.SqlTypes.SqlGuid.Value** is  
 19 equal to the **Guid** parameter. A **System.Guid**.

20 **op\_Inequality**

21  
 22 [C#] public static SqlBoolean operator !=(SqlGuid x, SqlGuid y);  
 23 [C++] public: static SqlBoolean op\_Inequality(SqlGuid x, SqlGuid y);  
 24 [VB] returnValue = SqlGuid.op\_Inequality(x, y)  
 25 [JScript] returnValue = x != y;

## Description

Performs a logical comparison on two **System.Data.SqlTypes.SqlGuid** structures to determine if they are equal.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **SqlGuid** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **SqlGuid** structure. A **SqlGuid** structure.

op\_LessThan

[C#] public static SqlBoolean operator

[C++] public: static SqlBoolean op\_LessThan(SqlGuid x, SqlGuid y);

[VB] returnValue = SqlGuid.op\_LessThan(x, y)

[JScript] returnValue = x < y;

## Description

Compares two instances of **System.Data.SqlTypes.SqlGuid** to determine if the first is less than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **SqlGuid** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the

**SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlGuid** structure. A **SqlGuid** structure.

**op\_LessThanOrEqual**

[C#] public static SqlBoolean operator <=(SqlGuid x, SqlGuid y);

[C++] public: static SqlBoolean op\_LessThanOrEqual(SqlGuid x, SqlGuid y);

[VB] returnValue = SqlGuid.op\_LessThanOrEqual(x, y)

[JScript] returnValue = x <= y;

### *Description*

Compares two instances of **System.Data.SqlTypes.SqlGuid** to determine if the first is less than or equal to the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **SqlGuid** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **SqlGuid** structure. A **SqlGuid** structure.

### **Parse**

[C#] public static SqlGuid Parse(string s);

[C++] public: static SqlGuid Parse(String\* s);

[VB] Public Shared Function Parse(ByVal s As String) As SqlGuid

[JScript] public static function Parse(s : String) : SqlGuid;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

[ . ] [ . ]  
ToByteArray

```
[C#]          public          byte[]          ToByteArray();
[C++]      public:      unsigned      char      ToByteArray()      __gc[];
[VB]      Public      Function      ToByteArray()      As      Byte()
[JScript]      public      function      ToByteArray()      :      Byte[];
```

*Description*

Converts this **System.Data.SqlTypes.SqlGuid** structure to a byte array.  
*Return Value:* An array of bytes representing the **System.Data.SqlTypes.SqlGuid.Value** of this **SqlGuid** structure.

ToSqlBinary

```
[C#]          public          SqlBinary          ToSqlBinary();
[C++]      public:          SqlBinary          ToSqlBinary();
[VB]      Public      Function      ToSqlBinary()      As      SqlBinary
[JScript]      public      function      ToSqlBinary()      :      SqlBinary;
```

*Description*

[ . ]  
ToSqlString



Represents a 16-bit signed integer to be stored in or retrieved from a database.

ToString

[C#]	public	static	readonly	SqlInt16	MaxValue;
[C++]	public:	static		SqlInt16	MaxValue;
[VB]	Public	Shared	ReadOnly	MaxValue	As SqlInt16
[JScript]	public	static	var	MaxValue	: SqlInt16;

*Description*

A constant representing the largest possible value of a **System.Data.SqlTypes.SqlInt16**.

The value of this constant is 32,767.

ToString

[C#]	public	static	readonly	SqlInt16	MinValue;
[C++]	public:	static		SqlInt16	MinValue;
[VB]	Public	Shared	ReadOnly	MinValue	As SqlInt16
[JScript]	public	static	var	MinValue	: SqlInt16;

*Description*

A constant representing the smallest possible value of a **System.Data.SqlTypes.SqlInt16**.

The value of this constant is -32,768.

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      static      readonly      SqlInt16      Null;
[C++]      public:      static      SqlInt16      Null;
[VB]      Public      Shared      ReadOnly      Null      As      SqlInt16
[JScript]      public      static      var      Null      :      SqlInt16;
```

*Description*

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlInt16.Value** property of an instance of the **System.Data.SqlTypes.SqlInt16** structure.

**System.Data.SqlTypes.SqlInt16.Null** functions as a constant for the **System.Data.SqlTypes.SqlInt16** structure.

**ToString**

```
[C#]      public      static      readonly      SqlInt16      Zero;
[C++]      public:      static      SqlInt16      Zero;
[VB]      Public      Shared      ReadOnly      Zero      As      SqlInt16
[JScript]      public      static      var      Zero      :      SqlInt16;
```

*Description*

Represents a zero value that can be assigned to the **System.Data.SqlTypes.SqlInt16.Value** property of an instance of the **System.Data.SqlTypes.SqlInt16** structure.

The **System.Data.SqlTypes.SqlInt16.Zero** field is a constant for the **System.Data.SqlTypes.SqlInt16** structure.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

SqlInt16

*Example Syntax:*

ToString

[C#]	public	SqlInt16(short	value);
[C++]	public:	SqlInt16(short	value);
[VB]	Public	Sub	New(ByVal value As Short)
[JScript]	public	function	SqlInt16(value : Int16);

*Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlInt16** structure using the supplied short integer parameter. A short integer.

IsNull

ToString

[C#]	public	bool	IsNull	{get;}
[C++]	public:	__property	bool	get_IsNull();
[VB]	Public	ReadOnly	Property	IsNull As Boolean
[JScript]	public	function	get	IsNull() : Boolean;

*Description*

Indicates whether or not **System.Data.SqlTypes.SqlInt16.Value** is null.

Value

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]          public          short          Value          {get;}
[C++]          public:          __property          short          get_Value();
[VB]          Public          ReadOnly          Property          Value          As          Short
[JScript]          public          function          get          Value()          :          Int16;
```

*Description*

Gets the value of this instance of **System.Data.SqlTypes.SqlInt16** structure. This property is read-only.

Add

```
[C#]          public          static          SqlInt16          Add(SqlInt16          x,          SqlInt16          y);
[C++]          public:          static          SqlInt16          Add(SqlInt16          x,          SqlInt16          y);
[VB]          Public          Shared          Function          Add(ByVal          x          As          SqlInt16,          ByVal          y          As          SqlInt16)          As          SqlInt16
[JScript]          public          static          function          Add(x          :          SqlInt16,          y          :          SqlInt16)          :          SqlInt16;
```

*Description*

[ .]

BitwiseAnd

```
[C#]          public          static          SqlInt16          BitwiseAnd(SqlInt16          x,          SqlInt16          y);
[C++]          public:          static          SqlInt16          BitwiseAnd(SqlInt16          x,          SqlInt16          y);
[VB]          Public          Shared          Function          BitwiseAnd(ByVal          x          As          SqlInt16,          ByVal          y          As          SqlInt16)          As          SqlInt16
```



*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

### Divide

```
[C#]    public    static    SqlInt16    Divide(SqlInt16    x,    SqlInt16    y);
[C++]    public:    static    SqlInt16    Divide(SqlInt16    x,    SqlInt16    y);
[VB] Public Shared Function Divide(ByVal x As SqlInt16, ByVal y As SqlInt16)
As
                                SqlInt16
[JScript] public static function Divide(x : SqlInt16, y : SqlInt16) : SqlInt16;
```

### Description

[ . ]

### Equals

```
[C#]    public    override    bool    Equals(object    value);
[C++]    public:    bool    Equals(Object*    value);
[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean
[JScript] public override function Equals(value : Object) : Boolean;
```

### Description

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlInt32.Value** property of the **System.Data.SqlTypes.SqlInt16** object.

*Return Value:* **true** if object is an instance of **System.Data.SqlTypes.SqlInt16** and the two are equal; otherwise **false** . The object to be compared.

## Equals

```
[C#] public static new SqlBoolean Equals(SqlInt16 x, SqlInt16 y);
[C++] public: static SqlBoolean Equals(SqlInt16 x, SqlInt16 y);
[VB] Shadows Public Shared Function Equals(ByVal x As SqlInt16, ByVal y As
SqlInt16) As SqlBoolean
[JScript] public static hide function Equals(x : SqlInt16, y : SqlInt16) :
SqlBoolean;
```

### Description

[ .]

### GetHashCode

```
[C#] public override int GetHashCode();
[C++] public: int GetHashCode();
[VB] Overrides Public Function GetHashCode() As Integer
[JScript] public override function GetHashCode() : int;
```

### Description

Returns the hash code for this instance.

*Return Value:* A 32-bit signed integer hash code.

### GreaterThan

```
[C#] public static SqlBoolean GreaterThan(SqlInt16 x, SqlInt16 y);
[C++] public: static SqlBoolean GreaterThan(SqlInt16 x, SqlInt16 y);
```

1 [VB] Public Shared Function GreaterThan(ByVal x As SqlInt16, ByVal y As  
2 SqlInt16) As SqlBoolean  
3 [JScript] public static function GreaterThan(x : SqlInt16, y : SqlInt16) :  
4 SqlBoolean;

5  
6 *Description*

7 [ .]  
8 GreaterThanOrEqualTo

9  
10 [C#] public static SqlBoolean GreaterThanOrEqualTo(SqlInt16 x, SqlInt16 y);  
11 [C++] public: static SqlBoolean GreaterThanOrEqualTo(SqlInt16 x, SqlInt16 y);  
12 [VB] Public Shared Function GreaterThanOrEqualTo(ByVal x As SqlInt16, ByVal y  
13 As SqlInt16) As SqlBoolean  
14 [JScript] public static function GreaterThanOrEqualTo(x : SqlInt16, y : SqlInt16) :  
15 SqlBoolean;

16  
17 *Description*

18 [ .]  
19 LessThan

20  
21 [C#] public static SqlBoolean LessThan(SqlInt16 x, SqlInt16 y);  
22 [C++] public: static SqlBoolean LessThan(SqlInt16 x, SqlInt16 y);  
23 [VB] Public Shared Function LessThan(ByVal x As SqlInt16, ByVal y As  
24 SqlInt16) As SqlBoolean  
25 [JScript] public static function LessThan(x : SqlInt16, y : SqlInt16) : SqlBoolean;

*Description*

[ .]

LessThanOrEqual

[C#] public static SqlBoolean LessThanOrEqual(SqlInt16 x, SqlInt16 y);

[C++] public: static SqlBoolean LessThanOrEqual(SqlInt16 x, SqlInt16 y);

[VB] Public Shared Function LessThanOrEqual(ByVal x As SqlInt16, ByVal y As

SqlInt16) As SqlBoolean

[JScript] public static function LessThanOrEqual(x : SqlInt16, y : SqlInt16) :

SqlBoolean;

*Description*

[ .]

Mod

[C#] public static SqlInt16 Mod(SqlInt16 x, SqlInt16 y);

[C++] public: static SqlInt16 Mod(SqlInt16 x, SqlInt16 y);

[VB] Public Shared Function Mod(ByVal x As SqlInt16, ByVal y As SqlInt16) As

SqlInt16

[JScript] public static function Mod(x : SqlInt16, y : SqlInt16) : SqlInt16;

*Description*

[ .]

Multiply

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public static SqlInt16 Multiply(SqlInt16 x, SqlInt16 y);
[C++] public: static SqlInt16 Multiply(SqlInt16 x, SqlInt16 y);
[VB] Public Shared Function Multiply(ByVal x As SqlInt16, ByVal y As
SqlInt16) As SqlInt16
[JScript] public static function Multiply(x : SqlInt16, y : SqlInt16) : SqlInt16;
```

*Description*

[ .]  
NotEquals

```
[C#] public static SqlBoolean NotEquals(SqlInt16 x, SqlInt16 y);
[C++] public: static SqlBoolean NotEquals(SqlInt16 x, SqlInt16 y);
[VB] Public Shared Function NotEquals(ByVal x As SqlInt16, ByVal y As
SqlInt16) As SqlBoolean
[JScript] public static function NotEquals(x : SqlInt16, y : SqlInt16) : SqlBoolean;
```

*Description*

[ .]  
OnesComplement

```
[C#] public static SqlInt16 OnesComplement(SqlInt16 x);
[C++] public: static SqlInt16 OnesComplement(SqlInt16 x);
[VB] Public Shared Function OnesComplement(ByVal x As SqlInt16) As
SqlInt16
```

[JScript] public static function OnesComplement(x : SqlInt16) : SqlInt16;

*Description*

[ .]

op\_Addition

[C#] public static SqlInt16 operator +(SqlInt16 x, SqlInt16 y);

[C++] public: static SqlInt16 op\_Addition(SqlInt16 x, SqlInt16 y);

[VB] returnValue = SqlInt16.op\_Addition(x, y)

[JScript] returnValue = x + y;

*Description*

Computes the sum of the two **System.Data.SqlTypes.SqlInt16** operands.

**Return Value:** A **System.Data.SqlTypes.SqlInt16** structure whose **System.Data.SqlTypes.SqlInt16.Value** property contains the sum of the two **System.Data.SqlTypes.SqlInt16** operands. A **System.Data.SqlTypes.SqlInt16** structure. A **System.Data.SqlTypes.SqlInt16** structure.

op\_BitwiseAnd

[C#] public static SqlInt16 operator &(amp;SqlInt16 x, SqlInt16 y);

[C++] public: static SqlInt16 op\_BitwiseAnd(SqlInt16 x, SqlInt16 y);

[VB] returnValue = SqlInt16.op\_BitwiseAnd(x, y)

[JScript] returnValue = x & y;

*Description*

Computes the bitwise AND of its **System.Data.SqlTypes.SqlInt16** operands. A **System.Data.SqlTypes.SqlInt16** structure. A **System.Data.SqlTypes.SqlInt16** structure.

op\_BitwiseOr

[C#] public static SqlInt16 operator |(SqlInt16 x, SqlInt16 y);  
 [C++] public: static SqlInt16 op\_BitwiseOr(SqlInt16 x, SqlInt16 y);  
 [VB] returnValue = SqlInt16.op\_BitwiseOr(x, y)  
 [JScript] returnValue = x | y;

#### Description

Computes the bitwise OR of its two **System.Data.SqlTypes.SqlInt16** operands. A **System.Data.SqlTypes.SqlInt16** structure. A **System.Data.SqlTypes.SqlInt16** structure.

op\_Division

[C#] public static SqlInt16 operator /(SqlInt16 x, SqlInt16 y);  
 [C++] public: static SqlInt16 op\_Division(SqlInt16 x, SqlInt16 y);  
 [VB] returnValue = SqlInt16.op\_Division(x, y)  
 [JScript] returnValue = x / y;

#### Description

The division operator divides the first **System.Data.SqlTypes.SqlInt16** operand by the second.

*Return Value:* A **System.Data.SqlTypes.SqlInt16** whose

1 **System.Data.SqlTypes.SqlInt16.Value** property contains the results of the  
 2 division. A **System.Data.SqlTypes.SqlInt16** structure. A  
 3 **System.Data.SqlTypes.SqlInt16** structure.

4 **op\_Equality**

5  
 6 [C#] public static SqlBoolean operator ==(SqlInt16 x, SqlInt16 y);  
 7 [C++] public: static SqlBoolean op\_Equality(SqlInt16 x, SqlInt16 y);  
 8 [VB] returnValue = SqlInt16.op\_Equality(x, y)  
 9 [JScript] returnValue = x == y;

# 11 *Description*

12 Performs a logical comparison of two **System.Data.SqlTypes.SqlInt16**  
 13 structures to determine if they are equal.

14 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 15 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or  
 16 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If  
 17 either instance of **System.Data.SqlTypes.SqlInt16** is null, the  
 18 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 19 **System.Data.SqlTypes.SqlBoolean** will be  
 20 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt16**  
 21 structure. A **System.Data.SqlTypes.SqlInt16** structure.

22 **op\_ExclusiveOr**

23  
 24 [C#] public static SqlInt16 operator ^(SqlInt16 x, SqlInt16 y);  
 25 [C++] public: static SqlInt16 op\_ExclusiveOr(SqlInt16 x, SqlInt16 y);

1	[VB]	returnValue	=	SqlInt16.op_ExclusiveOr(x,	y)
2	[JScript]	returnValue	=	x	^ y;

#### 4 Description

5        Performs a bitwise exclusive-OR operation on the supplied parameters. A  
6        **System.Data.SqlTypes.SqlInt16** structure. A **System.Data.SqlTypes.SqlInt16**  
7        structure.

8	op_Explicit
---	-------------

```
10 || [C#] public static explicit operator SqlInt16(SqlBoolean x);
```

```
11 || [C++] public: static SqlInt16 op_Explicit(SqlBoolean x);
```

12		[VB]	returnValue	=	SqlInt16.op_Explicit(x)
----	--	------	-------------	---	-------------------------

```
13 [JScript] returnValue = SqlInt16(x);
```

15 *Description*

Converts the supplied **System.Data.SqlTypes.SqlBit** structure to  
**System.Data.SqlTypes.SqlInt16**.

18 *Return Value:* A new **System.Data.SqlTypes.SqlInt16** structure whose  
19 **System.Data.SqlTypes.SqlInt16.Value** property is equal to the  
20 **System.Data.SqlTypes.SqlBit.ByteValue** property of the  
21 **System.Data.SqlTypes.SqlBit** parameter. A **System.Data.SqlTypes.SqlBit**  
22 structure.

23 | op Explicit

```
25 [C#] public static explicit operator SqlInt16(SqlDecimal x);
```

```

[C++]      public:      static      SqlInt16      op_Explicit(SqlDecimal      x);
[VB]              returnValue      =                  SqlInt16.op_Explicit(x)
[JScript]              returnValue      =                  SqlInt16(x);
    
```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlDecimal** structure to **System.Data.SqlTypes.SqlInt16**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt16** structure whose **System.Data.SqlTypes.SqlInt16.Value** property is equal to the **System.Data.SqlTypes.SqlDecimal.Value** property of the **System.Data.SqlTypes.SqlDecimal** parameter. A **System.Data.SqlTypes.SqlDecimal** structure.

#### *op\_Explicit*

```

[C#]      public      static      explicit      operator      SqlInt16(SqlDouble      x);
[C++]      public:      static      SqlInt16      op_Explicit(SqlDouble      x);
[VB]              returnValue      =                  SqlInt16.op_Explicit(x)
[JScript]              returnValue      =                  SqlInt16(x);
    
```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlDouble** structure to **System.Data.SqlTypes.SqlInt16**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt16** structure whose **System.Data.SqlTypes.SqlInt16.Value** property is equal to the integer portion of

1 the **System.Data.SqlTypes.SqlDouble** parameter. A  
2 **System.Data.SqlTypes.SqlDouble** structure.

3 op\_Explicit

4  
5 [C#] public static explicit operator short(SqlInt16 x);  
6 [C++] public: static short op\_Explicit();  
7 [VB] returnValue = SqlInt16.op\_Explicit(x)  
8 [JScript] returnValue = Int16(x);

9  
10 *Description*

11 Converts the supplied **System.Data.SqlTypes.SqlInt16** structure to a short  
12 integer.

13 *Return Value:* A short integer whose value is the Value of the  
14 **System.Data.SqlTypes.SqlInt16** parameter. A **System.Data.SqlTypes.SqlInt16**  
15 structure.

16 op\_Explicit

17  
18 [C#] public static explicit operator SqlInt16(SqlInt32 x);  
19 [C++] public: static SqlInt16 op\_Explicit(SqlInt32 x);  
20 [VB] returnValue = SqlInt16.op\_Explicit(x)  
21 [JScript] returnValue = SqlInt16(x);

22  
23 *Description*

24 Converts the supplied **System.Data.SqlTypes.SqlInt32** structure to  
25 **System.Data.SqlTypes.SqlInt16**

1 *Return Value:* A new **System.Data.SqlTypes.SqlInt16** structure whose  
 2 **System.Data.SqlTypes.SqlInt16.Value** property is equal to the  
 3 **System.Data.SqlTypes.SqlInt32.Value** of the supplied  
 4 **System.Data.SqlTypes.SqlInt32** parameter. A **System.Data.SqlTypes.SqlInt32**  
 5 structure.

6 **op\_Explicit**

7  
 8 [C#] public static explicit operator SqlInt16(SqlInt64 x);

9 [C++] public: static SqlInt16 op\_Explicit(SqlInt64 x);

10 [VB] returnValue = SqlInt16.op\_Explicit(x)

11 [JScript] returnValue = SqlInt16(x);

### 13 *Description*

14 Converts the supplied **System.Data.SqlTypes.SqlInt64** structure to  
 15 **System.Data.SqlTypes.SqlInt16**

16 *Return Value:* A new **System.Data.SqlTypes.SqlInt16** structure whose  
 17 **System.Data.SqlTypes.SqlInt16.Value** property is equal to the  
 18 **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64**  
 19 parameter. A **System.Data.SqlTypes.SqlInt64** structure.

20 **op\_Explicit**

21  
 22 [C#] public static explicit operator SqlInt16(SqlMoney x);

23 [C++] public: static SqlInt16 op\_Explicit(SqlMoney x);

24 [VB] returnValue = SqlInt16.op\_Explicit(x)

25 [JScript] returnValue = SqlInt16(x);

[illegible]

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

```

1
2 [C#]    public    static    explicit    operator    SqlInt16(SqlString    x);
3 [C++]    public:    static    SqlInt16    op_Explicit(SqlString    x);
4 [VB]        returnValue    =    SqlInt16.op_Explicit(x)
5 [JScript]        returnValue    =    SqlInt16(x);

```

#### 6 *Description*

7 Converts the supplied **System.Data.SqlTypes.SqlString** object to  
8 **System.Data.SqlTypes.SqlInt16**.

9 *Return Value:* A new **System.Data.SqlTypes.SqlInt16** structure whose  
10 **System.Data.SqlTypes.SqlInt16.Value** property is equal to the value represented  
11 by the **System.Data.SqlTypes.SqlString** object parameter. A  
12 **System.Data.SqlTypes.SqlString** object.  
13

14 **op\_GreaterThan**

```

15
16 [C#]    public    static    SqlBoolean    operator    >(SqlInt16    x,    SqlInt16    y);
17 [C++]    public:    static    SqlBoolean    op_GreaterThan(SqlInt16    x,    SqlInt16    y);
18 [VB]        returnValue    =    SqlInt16.op_GreaterThan(x,    y)
19 [JScript]        returnValue    =    x    >    y;

```

#### 20 *Description*

21 Compares two instances of **System.Data.SqlTypes.SqlInt16** to determine  
22 if the first is greater than the second.

23 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
24 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the  
25

1 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
 2 instance of **System.Data.SqlTypes.SqlInt16** is null, the  
 3 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 4 **System.Data.SqlTypes.SqlBoolean** will be  
 5 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt16**  
 6 structure. A **System.Data.SqlTypes.SqlInt16** structure.

7 **op\_GreaterThanOrEqual**

8  
 9 [C#] public static SqlBoolean operator >=(SqlInt16 x, SqlInt16 y);  
 10 [C++] public: static SqlBoolean op\_GreaterThanOrEqual(SqlInt16 x, SqlInt16 y);  
 11 [VB] returnValue = SqlInt16.op\_GreaterThanOrEqual(x, y)  
 12 [JScript] returnValue = x >= y;

13  
 14 *Description*

15 Compares two **System.Data.SqlTypes.SqlInt16** structures to determine if  
 16 the first is greater than or equal to the second.

17 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 18 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greaater than or  
 19 equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**  
 20 . If either instance of **System.Data.SqlTypes.SqlInt16** is null, the  
 21 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 22 **System.Data.SqlTypes.SqlBoolean** will be  
 23 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt16**  
 24 structure. A **System.Data.SqlTypes.SqlInt16** structure.

25 **op\_Implicit**

```

1
2 [C#]      public      static      implicit      operator      SqlInt16(short      x);
3 [C++]     public:     static      SqlInt16      op_Implicit(short      x);
4 [VB]      returnValue      =      SqlInt16.op_Implicit(x)
5 [JScript]      returnValue      =      x;

```

6

7 *Description*

8 Converts the supplied short integer to **System.Data.SqlTypes.SqlInt16** . A

9 short integer value.

10 op\_Implicit

```

11
12 [C#]      public      static      implicit      operator      SqlInt16(SqlByte      x);
13 [C++]     public:     static      SqlInt16      op_Implicit(SqlByte      x);
14 [VB]      returnValue      =      SqlInt16.op_Implicit(x)
15 [JScript]      returnValue      =      x;

```

16

17 *Description*

18 Converts the supplied **System.Data.SqlTypes.SqlByte** structure to

19 **System.Data.SqlTypes.SqlInt16**

20 *Return Value:* A new **System.Data.SqlTypes.SqlInt16** structure whose

21 **System.Data.SqlTypes.SqlInt16.Value** property is equal to the

22 **System.Data.SqlTypes.SqlByte.Value** property of the

23 **System.Data.SqlTypes.SqlByte** parameter. A **System.Data.SqlTypes.SqlByte**

24 structure.

25 op\_Inequality

```

1
2 [C#] public static SqlBoolean operator !=(SqlInt16 x, SqlInt16 y);
3 [C++] public: static SqlBoolean op_Inequality(SqlInt16 x, SqlInt16 y);
4 [VB]     returnValue      =      SqlInt16.op_Inequality(x,      y)
5 [JScript]     returnValue      =      x      !=      y;
6

```

### *Description*

Performs a logical comparison of two **System.Data.SqlTypes.SqlInt16** structures to determine if they are equal.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **System.Data.SqlTypes.SqlInt16** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlInt16** structure. A **System.Data.SqlTypes.SqlInt16** structure.

op\_LessThan

```

18
19
20 [C#]     public     static     SqlBoolean     operator
21 [C++] public: static SqlBoolean op_LessThan(SqlInt16 x, SqlInt16 y);
22 [VB]     returnValue      =      SqlInt16.op_LessThan(x,      y)
23 [JScript]     returnValue      =      x      <      y;
24

```

### *Description*

Compares two instances of **System.Data.SqlTypes.SqlInt16** to determine if the first is less than the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **System.Data.SqlTypes.SqlInt16** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt16** structure. A **System.Data.SqlTypes.SqlInt16** structure.

**op\_LessThanOrEqual**

[C#] public static SqlBoolean operator <=(SqlInt16 x, SqlInt16 y);

[C++] public: static SqlBoolean op\_LessThanOrEqual(SqlInt16 x, SqlInt16 y);

[VB] returnValue = SqlInt16.op\_LessThanOrEqual(x, y)

[JScript] returnValue = x <= y;

### Description

Compares two **System.Data.SqlTypes.SqlInt16** structures to determine if the first is less than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **System.Data.SqlTypes.SqlInt16** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the



1       The multiplication operator computes the product of the two  
2 **System.Data.SqlTypes.SqlInt16** parameters.

3 *Return Value:* A **System.Data.SqlTypes.SqlInt16** structure whose  
4 **System.Data.SqlTypes.SqlInt16.Value** contains the product of the two  
5 parameters. A **System.Data.SqlTypes.SqlInt16** structure. A  
6 **System.Data.SqlTypes.SqlInt16** structure.

7       op\_OnesComplement

9 [C#]       public       static       SqlInt16       operator       ~(SqlInt16       x);

10 [C++]       public:       static       SqlInt16       op\_OnesComplement(SqlInt16       x);

11 [VB]       returnValue       =       SqlInt16.op\_OnesComplement(x)

12 [JScript]       returnValue       =       ~x;

14 *Description*

15       The ~ operator performs a bitwise one's complement operation on its  
16 **System.Data.SqlTypes.SqlByte** operand. A **System.Data.SqlTypes.SqlInt16**  
17 structure.

18       op\_Subtraction

20 [C#]       public       static       SqlInt16       operator       -(SqlInt16       x,       SqlInt16       y);

21 [C++]       public:       static       SqlInt16       op\_Subtraction(SqlInt16       x,       SqlInt16       y);

22 [VB]       returnValue       =       SqlInt16.op\_Subtraction(x,       y)

23 [JScript]       returnValue       =       x       -       y;

25 *Description*

Subtracts the second **System.Data.SqlTypes.SqlInt16** parameter from the first.

*Return Value:* A **System.Data.SqlTypes.SqlInt16** structure whose **System.Data.SqlTypes.SqlInt16.Value** property contains the results of the subtraction. A **System.Data.SqlTypes.SqlInt16** structure. A **System.Data.SqlTypes.SqlInt16** structure.

op\_UnaryNegation

```
[C#]      public      static      SqlInt16      operator      -(SqlInt16      x);
[C++]     public:     static      SqlInt16      op_UnaryNegation(SqlInt16      x);
[VB]      returnValue      =      SqlInt16.op_UnaryNegation(x)
[JScript]      returnValue      =      -x;
```

#### *Description*

The unary minus operator negates the **System.Data.SqlTypes.SqlInt16.Value** of the **System.Data.SqlTypes.SqlInt16** operand. A **System.Data.SqlTypes.SqlInt16** structure.

Parse

```
[C#]      public      static      SqlInt16      Parse(string      s);
[C++]     public:     static      SqlInt16      Parse(String*      s);
[VB]     Public Shared Function Parse(ByVal s As String) As SqlInt16
[JScript] public static function Parse(s : String) : SqlInt16;
```

#### *Description*

```

1      [ .][ .]
2      Subtract
3
4  [C#]   public   static   SqlInt16   Subtract(SqlInt16   x,   SqlInt16   y);
5  [C++]  public:   static   SqlInt16   Subtract(SqlInt16   x,   SqlInt16   y);
6  [VB]   Public  Shared  Function Subtract(ByVal x As SqlInt16, ByVal y As
7  SqlInt16)                                     As                               SqlInt16
8  [JScript] public static function Subtract(x : SqlInt16, y : SqlInt16) : SqlInt16;
9

```

*Description*

```

11     [ .]
12     ToSqlBoolean
13
14  [C#]           public           SqlBoolean           ToSqlBoolean();
15  [C++]          public:          SqlBoolean           ToSqlBoolean();
16  [VB]   Public   Function   ToSqlBoolean()   As       SqlBoolean
17  [JScript]   public   function   ToSqlBoolean()   :       SqlBoolean;
18

```

*Description*

```

20     [ .]
21     ToSqlByte
22
23  [C#]           public           SqlByte           ToSqlByte();
24  [C++]          public:          SqlByte           ToSqlByte();
25  [VB]   Public   Function   ToSqlByte()   As       SqlByte

```

1 [JScript] public function ToSqlByte() : SqlByte;

2

3 *Description*

4 [ .]

5 ToSqlDecimal

6

7 [C#] public SqlDecimal ToSqlDecimal();

8 [C++] public: SqlDecimal ToSqlDecimal();

9 [VB] Public Function ToSqlDecimal() As SqlDecimal

10 [JScript] public function ToSqlDecimal() : SqlDecimal;

11

12 *Description*

13 [ .]

14 ToSqlDouble

15

16 [C#] public SqlDouble ToSqlDouble();

17 [C++] public: SqlDouble ToSqlDouble();

18 [VB] Public Function ToSqlDouble() As SqlDouble

19 [JScript] public function ToSqlDouble() : SqlDouble;

20

21 *Description*

22 [ .]

23 ToSqlInt32

24

25 [C#] public SqlInt32 ToSqlInt32();

1	[C++]	public:	SqlInt32	ToSqlInt32();
2	[VB]	Public	Function	ToSqlInt32() As SqlInt32
3	[JScript]	public	function	ToSqlInt32() : SqlInt32;
4				
5	<i>Description</i>			
6	[ .]			
7	ToSqlInt64			
8				
9	[C#]	public	SqlInt64	ToSqlInt64();
10	[C++]`	public:	SqlInt64	ToSqlInt64();
11	[VB]	Public	Function	ToSqlInt64() As SqlInt64
12	[JScript]	public	function	ToSqlInt64() : SqlInt64;
13				
14	<i>Description</i>			
15	[ .]			
16	ToSqlMoney			
17				
18	[C#]	public	SqlMoney	ToSqlMoney();
19	[C++]`	public:	SqlMoney	ToSqlMoney();
20	[VB]	Public	Function	ToSqlMoney() As SqlMoney
21	[JScript]	public	function	ToSqlMoney() : SqlMoney;
22				
23	<i>Description</i>			
24	[ .]			
25	ToSqlSingle			

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]          public          SqlSingle          ToSqlSingle();
[C++]          public:          SqlSingle          ToSqlSingle();
[VB]      Public      Function  ToSqlSingle()      As      SqlSingle
[JScript]      public      function  ToSqlSingle()      :      SqlSingle;
```

*Description*

[ .]  
ToSqlString

```
[C#]          public          SqlString          ToSqlString();
[C++]          public:          SqlString          ToSqlString();
[VB]      Public      Function  ToSqlString()      As      SqlString
[JScript]      public      function  ToSqlString()      :      SqlString;
```

*Description*

[ .]  
ToString

```
[C#]          public          override          string          ToString();
[C++]          public:          String*          ToString();
[VB]      Overrides      Public      Function  ToString()      As      String
[JScript]      public      override      function  ToString()      :      String; Converts a
System.Data.SqlTypes.SqlInt16      structure      to      System.String .
```

1  
2 *Description*

3 Converts a **System.Data.SqlTypes.SqlInt16** structure to **System.String** .

4 *Return Value:* A **System.String** object representing the  
5 **System.Data.SqlTypes.SqlInt16.Value** of this instance of  
6 **System.Data.SqlTypes.SqlInt16** .

7 Xor

8  
9 [C#] public static SqlInt16 Xor(SqlInt16 x, SqlInt16 y);

10 [C++] public: static SqlInt16 Xor(SqlInt16 x, SqlInt16 y);

11 [VB] Public Shared Function Xor(ByVal x As SqlInt16, ByVal y As SqlInt16) As  
12 SqlInt16

13 [JScript] public static function Xor(x : SqlInt16, y : SqlInt16) : SqlInt16;

14  
15 *Description*

16 [ .]

17 SqlInt32 structure (System.Data.SqlTypes)

18 Xor

19  
20  
21 *Description*

22 Represents a 32-bit signed integer to be stored in or retrieved from a  
23 database.

24 Xor  
25

```

1
2 [C#]      public      static      readonly      SqlInt32      MaxValue;
3 [C++]      public:      static      SqlInt32      MaxValue;
4 [VB]      Public      Shared      ReadOnly      MaxValue      As      SqlInt32
5 [JScript]  public      static      var      MaxValue      :      SqlInt32;

```

6

7 *Description*

8       A constant representing the largest possible value of a

9 **System.Data.SqlTypes.SqlInt32** .

10       The value for this constant is 2,147,483,647.

11       Xor

```

12
13 [C#]      public      static      readonly      SqlInt32      MinValue;
14 [C++]      public:      static      SqlInt32      MinValue;
15 [VB]      Public      Shared      ReadOnly      MinValue      As      SqlInt32
16 [JScript]  public      static      var      MinValue      :      SqlInt32;

```

17

18 *Description*

19       A constant representing the smallest possible value of a

20 **System.Data.SqlTypes.SqlInt32** .

21       The value of this constant is -2,147,483,648.

22       Xor

```

23
24 [C#]      public      static      readonly      SqlInt32      Null;
25 [C++]      public:      static      SqlInt32      Null;

```

1 [VB] Public Shared ReadOnly Null As SqlInt32  
 2 [JScript] public static var Null : SqlInt32;

3  
 4 *Description*

5 Represents a null value that can be assigned to the  
 6 **System.Data.SqlTypes.SqlInt32.Value** property of an instance of the  
 7 **System.Data.SqlTypes.SqlInt32** structure.

8 **System.Data.SqlTypes.SqlInt32.Null** functions as a constant for the  
 9 **System.Data.SqlTypes.SqlInt32** structure.

10 Xor

11  
 12 [C#] public static readonly SqlInt32 Zero;  
 13 [C++] public: static SqlInt32 Zero;  
 14 [VB] Public Shared ReadOnly Zero As SqlInt32  
 15 [JScript] public static var Zero : SqlInt32;

16  
 17 *Description*

18 Represents a zero value that can be assigned to the  
 19 **System.Data.SqlTypes.SqlInt32.Value** property of an instance of the  
 20 **System.Data.SqlTypes.SqlInt32** structure.

21 The **System.Data.SqlTypes.SqlInt32.Zero** field is a constant for the  
 22 **System.Data.SqlTypes.SqlInt32** structure.

23 SqlInt32

24 *Example Syntax:*

25 Xor

```

1
2 [C#]          public          SqlInt32(int          value);
3 [C++]          public:          SqlInt32(int          value);
4 [VB]    Public    Sub    New(ByVal    value    As    Integer)
5 [JScript]    public    function    SqlInt32(value    :    int);

```

```

6
7 Description
8     Initializes a new instance of the System.Data.SqlTypes.SqlInt32 structure
9 using the supplied integer value.
10     IsNull
11     Xor

```

```

12
13 [C#]          public          bool          IsNull          {get;}
14 [C++]          public:          __property          bool          get_IsNull();
15 [VB]    Public    ReadOnly    Property    IsNull    As    Boolean
16 [JScript]    public    function    get    IsNull()    :    Boolean;

```

```

17
18 Description
19     Indicates whether or not System.Data.SqlTypes.SqlInt32.Value is null.
20     Value
21     Xor

```

```

22
23 [C#]          public          int          Value          {get;}
24 [C++]          public:          __property          int          get_Value();
25 [VB]    Public    ReadOnly    Property    Value    As    Integer

```

1 [JScript] public function get Value() : int;

2

3 *Description*

4 Gets the value of this **System.Data.SqlTypes.SqlInt32** structure. This  
5 property is read-only.

6 Add

7

8 [C#] public static SqlInt32 Add(SqlInt32 x, SqlInt32 y);

9 [C++] public: static SqlInt32 Add(SqlInt32 x, SqlInt32 y);

10 [VB] Public Shared Function Add(ByVal x As SqlInt32, ByVal y As SqlInt32) As  
11 SqlInt32

12 [JScript] public static function Add(x : SqlInt32, y : SqlInt32) : SqlInt32;

13

14 *Description*

15 [ .]

16 BitwiseAnd

17

18 [C#] public static SqlInt32 BitwiseAnd(SqlInt32 x, SqlInt32 y);

19 [C++] public: static SqlInt32 BitwiseAnd(SqlInt32 x, SqlInt32 y);

20 [VB] Public Shared Function BitwiseAnd(ByVal x As SqlInt32, ByVal y As  
21 SqlInt32) As SqlInt32

22 [JScript] public static function BitwiseAnd(x : SqlInt32, y : SqlInt32) : SqlInt32;

23

24 *Description*

25 [ .]

```

1      BitwiseOr
2
3  [C#]   public   static   SqlInt32   BitwiseOr(SqlInt32   x,   SqlInt32   y);
4  [C++]  public:   static   SqlInt32   BitwiseOr(SqlInt32   x,   SqlInt32   y);
5  [VB]   Public Shared Function BitwiseOr(ByVal x As SqlInt32, ByVal y As
6  SqlInt32)                                     As                               SqlInt32
7  [JScript] public static function BitwiseOr(x : SqlInt32, y : SqlInt32) : SqlInt32;

```

### 9 *Description*

10 [ .]

### 11 *CompareTo*

```

12
13 [C#]           public           int           CompareTo(object           value);
14 [C++]          public:          __sealed      int           CompareTo(Object*          value);
15 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
16 Integer
17 [JScript]      public   function   CompareTo(value   :   Object)   :   int;

```

### 19 *Description*

20 Compares this instance to the supplied object and returns an indication of  
21 their relative values.

22 *Return Value:* A signed number indicating the relative values of the instance and  
23 the object. The object to be compared.

### 24 *Divide*

25

```

1
2 [C#] public static SqlInt32 Divide(SqlInt32 x, SqlInt32 y);
3 [C++] public: static SqlInt32 Divide(SqlInt32 x, SqlInt32 y);
4 [VB] Public Shared Function Divide(ByVal x As SqlInt32, ByVal y As SqlInt32)
5 As SqlInt32
6 [JScript] public static function Divide(x : SqlInt32, y : SqlInt32) : SqlInt32;
7

```

#### *Description*

```

9 [ .]
10 Equals
11

```

```

12 [C#] public override bool Equals(object value);
13 [C++] public: bool Equals(Object* value);
14 [VB] Overrides Public Function Equals(ByVal value As Object) As Boolean
15 [JScript] public override function Equals(value : Object) : Boolean;
16

```

#### *Description*

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlInt32.Value** property of the **System.Data.SqlTypes.SqlInt32** object.

*Return Value:* **true** if object is an instance of **System.Data.SqlTypes.SqlInt32** and the two are equal; otherwise **false** . The object to be compared.

```

23 Equals
24

```

```

25 [C#] public static new SqlBoolean Equals(SqlInt32 x, SqlInt32 y);

```

```

1  [C++] public: static SqlBoolean Equals(SqlInt32 x, SqlInt32 y);
2  [VB] Shadows Public Shared Function Equals(ByVal x As SqlInt32, ByVal y As
3  SqlInt32) As SqlBoolean
4  [JScript] public static hide function Equals(x : SqlInt32, y : SqlInt32) :
5  SqlBoolean;

```

### Description

$$[\cdot]$$

## GetHashCode

11	[C#]	public	override	int	GetHashCode();
12	[C++]	public:		int	GetHashCode();
13	[VB]	Overrides	Public	Function	GetHashCode() As Integer
14	[JScript]	public	override	function	GetHashCode() : int;

### Description

Returns the hash code for this instance.

**Return Value:** A 32-bit signed integer hash code.

## GreaterThan

```

21 [C#] public static SqlBoolean GreaterThan(SqlInt32 x, SqlInt32 y);
22 [C++] public: static SqlBoolean GreaterThan(SqlInt32 x, SqlInt32 y);
23 [VB] Public Shared Function GreaterThan(ByVal x As SqlInt32, ByVal y As
24 SqlInt32) As SqlBoolean
25 [JScript] public static function GreaterThan(x : SqlInt32, y : SqlInt32) :

```

1 SqlBoolean;

2

3 *Description*

4 [ .]

5 GreaterThanOrEqualTo

6

7 [C#] public static SqlBoolean GreaterThanOrEqualTo(SqlInt32 x, SqlInt32 y);

8 [C++] public: static SqlBoolean GreaterThanOrEqualTo(SqlInt32 x, SqlInt32 y);

9 [VB] Public Shared Function GreaterThanOrEqualTo(ByVal x As SqlInt32, ByVal y

10 As SqlInt32) As SqlBoolean

11 [JScript] public static function GreaterThanOrEqualTo(x : SqlInt32, y : SqlInt32) :

12 SqlBoolean;

13

14 *Description*

15 [ .]

16 LessThan

17

18 [C#] public static SqlBoolean LessThan(SqlInt32 x, SqlInt32 y);

19 [C++] public: static SqlBoolean LessThan(SqlInt32 x, SqlInt32 y);

20 [VB] Public Shared Function LessThan(ByVal x As SqlInt32, ByVal y As

21 SqlInt32) As SqlBoolean

22 [JScript] public static function LessThan(x : SqlInt32, y : SqlInt32) : SqlBoolean;

23

24 *Description*

25 [ .]

## LessThanOrEqual

```
[C#] public static SqlBoolean LessThanOrEqual(SqlInt32 x, SqlInt32 y);  
[C++] public: static SqlBoolean LessThanOrEqual(SqlInt32 x, SqlInt32 y);  
[VB] Public Shared Function LessThanOrEqual(ByVal x As SqlInt32, ByVal y As  
SqlInt32) As SqlBoolean  
[JScript] public static function LessThanOrEqual(x : SqlInt32, y : SqlInt32) :  
SqlBoolean;
```

### *Description*

[ . ]

Mod

```
[C#] public static SqlInt32 Mod(SqlInt32 x, SqlInt32 y);  
[C++] public: static SqlInt32 Mod(SqlInt32 x, SqlInt32 y);  
[VB] Public Shared Function Mod(ByVal x As SqlInt32, ByVal y As SqlInt32) As  
SqlInt32  
[JScript] public static function Mod(x : SqlInt32, y : SqlInt32) : SqlInt32;
```

### *Description*

[ . ]

Multiply

```
[C#] public static SqlInt32 Multiply(SqlInt32 x, SqlInt32 y);  
[C++] public: static SqlInt32 Multiply(SqlInt32 x, SqlInt32 y);
```

1 [VB] Public Shared Function Multiply(ByVal x As SqlInt32, ByVal y As  
2 SqlInt32) As SqlInt32

3 [JScript] public static function Multiply(x : SqlInt32, y : SqlInt32) : SqlInt32;

4  
5 *Description*

6 [ .]

7 NotEquals

8  
9 [C#] public static SqlBoolean NotEquals(SqlInt32 x, SqlInt32 y);

10 [C++] public: static SqlBoolean NotEquals(SqlInt32 x, SqlInt32 y);

11 [VB] Public Shared Function NotEquals(ByVal x As SqlInt32, ByVal y As  
12 SqlInt32) As SqlBoolean

13 [JScript] public static function NotEquals(x : SqlInt32, y : SqlInt32) : SqlBoolean;

14  
15 *Description*

16 [ .]

17 OnesComplement

18  
19 [C#] public static SqlInt32 OnesComplement(SqlInt32 x);

20 [C++] public: static SqlInt32 OnesComplement(SqlInt32 x);

21 [VB] Public Shared Function OnesComplement(ByVal x As SqlInt32) As  
22 SqlInt32

23 [JScript] public static function OnesComplement(x : SqlInt32) : SqlInt32;

24  
25 *Description*

[ .]

op\_Addition

[C#] public static SqlInt32 operator +(SqlInt32 x, SqlInt32 y);

[C++] public: static SqlInt32 op\_Addition(SqlInt32 x, SqlInt32 y);

[VB] returnValue = SqlInt32.op\_Addition(x, y)

[JScript] returnValue = x + y;

### *Description*

The addition operator computes the sum of the two **System.Data.SqlTypes.SqlInt32** operands.

*Return Value:* A **System.Data.SqlTypes.SqlInt32** structure whose **System.Data.SqlTypes.SqlInt32.Value** property contains the sum of the two **System.Data.SqlTypes.SqlInt32** operands. A **System.Data.SqlTypes.SqlInt32** structure. A **System.Data.SqlTypes.SqlInt32** structure.

op\_BitwiseAnd

[C#] public static SqlInt32 operator &(amp;SqlInt32 x, SqlInt32 y);

[C++] public: static SqlInt32 op\_BitwiseAnd(SqlInt32 x, SqlInt32 y);

[VB] returnValue = SqlInt32.op\_BitwiseAnd(x, y)

[JScript] returnValue = x & y;

### *Description*

1        Computes the bitwise AND of its **System.Data.SqlTypes.SqlInt32**  
 2 operands.        A        **System.Data.SqlTypes.SqlInt32**        structure.        A  
 3 **System.Data.SqlTypes.SqlInt32** structure.

4        op\_BitwiseOr

6 [C#]    public    static    SqlInt32    operator    |(SqlInt32    x,    SqlInt32    y);  
 7 [C++]    public:    static    SqlInt32    op\_BitwiseOr(SqlInt32    x,    SqlInt32    y);  
 8 [VB]        returnValue        =        SqlInt32.op\_BitwiseOr(x,        y)  
 9 [JScript]        returnValue        =        x        |        y;

11        *Description*

12        Computes the bitwise OR of its two **System.Data.SqlTypes.SqlInt32**  
 13 operands.        A        **System.Data.SqlTypes.SqlInt32**        structure.        A  
 14 **System.Data.SqlTypes.SqlInt32** structure.

15        op\_Division

17 [C#]    public    static    SqlInt32    operator    /(SqlInt32    x,    SqlInt32    y);  
 18 [C++]    public:    static    SqlInt32    op\_Division(SqlInt32    x,    SqlInt32    y);  
 19 [VB]        returnValue        =        SqlInt32.op\_Division(x,        y)  
 20 [JScript]        returnValue        =        x        /        y;

22        *Description*

23        The division operator divides the first **System.Data.SqlTypes.SqlInt32**  
 24 parameter        from        the        second.

25 *Return Value:*        A        **System.Data.SqlTypes.SqlInt32**        whose

1 **System.Data.SqlTypes.SqlInt32.Value** property contains the results of the  
 2 division. A **System.Data.SqlTypes.SqlInt32** structure. A  
 3 **System.Data.SqlTypes.SqlInt32** structure.

4 **op\_Equality**

5  
 6 [C#] public static SqlBoolean operator ==(SqlInt32 x, SqlInt32 y);  
 7 [C++] public: static SqlBoolean op\_Equality(SqlInt32 x, SqlInt32 y);  
 8 [VB] returnValue = SqlInt32.op\_Equality(x, y)  
 9 [JScript] returnValue = x == y;

10  
 11 *Description*

12 Performs a logical comparison of the two **System.Data.SqlTypes.SqlInt32**  
 13 parameters to determine if they are equal.

14 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 15 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or  
 16 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If  
 17 either instance of **System.Data.SqlTypes.SqlInt32** is null, the  
 18 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 19 **System.Data.SqlTypes.SqlBoolean** will be  
 20 **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlInt32**  
 21 structure. A **System.Data.SqlTypes.SqlInt32** structure.

22 **op\_ExclusiveOr**

23  
 24 [C#] public static SqlInt32 operator ^(SqlInt32 x, SqlInt32 y);  
 25 [C++] public: static SqlInt32 op\_ExclusiveOr(SqlInt32 x, SqlInt32 y);

```

1 [VB]         returnValue      =      SqlInt32.op_ExclusiveOr(x,      y)
2 [JScript]    returnValue      =      x      ^      y;

```

3

#### 4 *Description*

5 Performs a bitwise exclusive-OR operation on the supplied parameters. A  
6 **System.Data.SqlTypes.SqlInt32** structure. A **System.Data.SqlTypes.SqlInt32**  
7 structure.

8 op\_Explicit

9

```

10 [C#]    public    static    explicit    operator    SqlInt32(SqlBoolean    x);

```

```

11 [C++]    public:    static    SqlInt32    op_Explicit(SqlBoolean    x);

```

```

12 [VB]         returnValue      =      SqlInt32.op_Explicit(x)

```

```

13 [JScript]    returnValue      =      SqlInt32(x);

```

14

#### 15 *Description*

16 Converts the supplied **System.Data.SqlTypes.SqlBit** to  
17 **System.Data.SqlTypes.SqlInt32**.

18 *Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose  
19 **System.Data.SqlTypes.SqlInt32.Value** property is equal to the  
20 **System.Data.SqlTypes.SqlBit.ByteValue** property of the  
21 **System.Data.SqlTypes.SqlBit** parameter. A **System.Data.SqlTypes.SqlBit**  
22 structure.

23 op\_Explicit

24

```

25 [C#]    public    static    explicit    operator    SqlInt32(SqlDecimal    x);

```

1	[C++]	public: static	SqlInt32	op_Explicit(SqlDecimal x);
2	[VB]	returnValue	=	SqlInt32.op_Explicit(x)
3	[JScript]	returnValue	=	SqlInt32(x);

## 5

6  
7

8  
9  
10  
11  
12

13

15  
16  
17  
18

## 20

21

22

23

24

1 **System.Data.SqlTypes.SqlDouble** parameter. A

2 **System.Data.SqlTypes.SqlDouble** structure.

3 op\_Explicit

4  
5 [C#] public static explicit operator int(SqlInt32 x);

6 [C++] public: static int op\_Explicit();

7 [VB] returnValue = SqlInt32.op\_Explicit(x)

8 [JScript] returnValue = Int32(x);

9  
10 *Description*

11 Converts the supplied **System.Data.SqlTypes.SqlInt32** structure to an  
12 integer. A **System.Data.SqlTypes.SqlInt32** structure.

13 op\_Explicit

14  
15 [C#] public static explicit operator SqlInt32(SqlInt64 x);

16 [C++] public: static SqlInt32 op\_Explicit(SqlInt64 x);

17 [VB] returnValue = SqlInt32.op\_Explicit(x)

18 [JScript] returnValue = SqlInt32(x);

19  
20 *Description*

21 Converts the supplied **System.Data.SqlTypes.SqlInt64** to  
22 **System.Data.SqlTypes.SqlInt32**.

23 *Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose

24 **System.Data.SqlTypes.SqlInt32.Value** property equals the

25 **System.Data.SqlTypes.SqlInt64.Value** property of the

1 **System.Data.SqlTypes.SqlInt64** parameter. A **System.Data.SqlTypes.SqlInt64**  
2 structure.

3 **op\_Explicit**

4  
5 [C#] public static explicit operator **SqlInt32**(**SqlMoney** x);

6 [C++] public: static **SqlInt32** op\_Explicit(**SqlMoney** x);

7 [VB] returnValue = **SqlInt32.op\_Explicit**(x)

8 [JScript] returnValue = **SqlInt32**(x);

9  
10 *Description*

11 Converts the supplied **System.Data.SqlTypes.SqlMoney** structure to  
12 **System.Data.SqlTypes.SqlInt32**

13 *Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose  
14 **System.Data.SqlTypes.SqlInt32.Value** property equals the  
15 **System.Data.SqlTypes.SqlMoney.Value** property of the  
16 **System.Data.SqlTypes.SqlMoney** parameter. A  
17 **System.Data.SqlTypes.SqlMoney** structure.

18 **op\_Explicit**

19  
20 [C#] public static explicit operator **SqlInt32**(**SqlSingle** x);

21 [C++] public: static **SqlInt32** op\_Explicit(**SqlSingle** x);

22 [VB] returnValue = **SqlInt32.op\_Explicit**(x)

23 [JScript] returnValue = **SqlInt32**(x);

24  
25 *Description*

Converts the supplied **System.Data.SqlTypes.SqlSingle** to **System.Data.SqlTypes.SqlInt32**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose **System.Data.SqlTypes.SqlInt32.Value** property equals the integer portion of the **System.Data.SqlTypes.SqlSingle** parameter. A **System.Data.SqlTypes.SqlSingle** structure.

op\_Explicit

[C#] public static explicit operator SqlInt32(SqlString x);

[C++] public: static SqlInt32 op\_Explicit(SqlString x);

[VB] returnValue = SqlInt32.op\_Explicit(x)

[JScript] returnValue = SqlInt32(x);

#### Description

Converts the supplied **System.Data.SqlTypes.SqlString** object to **System.Data.SqlTypes.SqlInt32**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose **System.Data.SqlTypes.SqlInt32.Value** property equals the value represented by the **System.Data.SqlTypes.SqlString** parameter. A **System.Data.SqlTypes.SqlString** object.

op\_GreaterThan

[C#] public static SqlBoolean operator >(SqlInt32 x, SqlInt32 y);

[C++] public: static SqlBoolean op\_GreaterThan(SqlInt32 x, SqlInt32 y);

[VB] returnValue = SqlInt32.op\_GreaterThan(x, y)

1 [JScript]           returnValue           =           x           >           y;

2

3 *Description*

4       Compares the two **System.Data.SqlTypes.SqlInt32** parameters to  
5 determine if the first is greater than the second.

6 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
7 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the  
8 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
9 instance of **System.Data.SqlTypes.SqlInt32** is null, the  
10 **System.Data.SqlTypes.SqlBoolean.Value** of the  
11 **System.Data.SqlTypes.SqlBoolean** will be  
12 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt32**  
13 structure. A **System.Data.SqlTypes.SqlInt32** structure.

14       op\_GreaterThanOrEqual

15

16 [C#] public static SqlBoolean operator >=(SqlInt32 x, SqlInt32 y);

17 [C++] public: static SqlBoolean op\_GreaterThanOrEqual(SqlInt32 x, SqlInt32 y);

18 [VB]       returnValue       =       SqlInt32.op\_GreaterThanOrEqual(x,       y)

19 [JScript]       returnValue       =       x       >=       y;

20

21 *Description*

22       Compares the two **System.Data.SqlTypes.SqlInt32** parameters to  
23 determine if the first is greater than or equal to the second.

24 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
25 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greaater than or

1 equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**  
 2 . If either instance of **System.Data.SqlTypes.SqlInt32** is null, the  
 3 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 4 **System.Data.SqlTypes.SqlBoolean** will be  
 5 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt32**  
 6 structure. A **System.Data.SqlTypes.SqlInt32** structure.

7 **op\_Implicit**

8  
 9 [C#] public static implicit operator SqlInt32(int x);  
 10 [C++] public: static SqlInt32 op\_Implicit(int x);  
 11 [VB] returnValue = SqlInt32.op\_Implicit(x)  
 12 [JScript] returnValue = x;

13  
 14 *Description*

15 Converts the supplied integer to **System.Data.SqlTypes.SqlInt32** .  
 16 *Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose Value  
 17 property is equal to the integer parameter. An integer value.

18 **op\_Implicit**

19  
 20 [C#] public static implicit operator SqlInt32(SqlByte x);  
 21 [C++] public: static SqlInt32 op\_Implicit(SqlByte x);  
 22 [VB] returnValue = SqlInt32.op\_Implicit(x)  
 23 [JScript] returnValue = x;

24  
 25 *Description*

Converts the supplied **System.Data.SqlTypes.SqlByte** property to **System.Data.SqlTypes.SqlInt32**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose **System.Data.SqlTypes.SqlInt32.Value** property equals the **System.Data.SqlTypes.SqlByte.Value** property of the **System.Data.SqlTypes.SqlByte** parameter. A **System.Data.SqlTypes.SqlByte** structure.

op\_Implicit

[C#] public static implicit operator SqlInt32(SqlInt16 x);

[C++] public: static SqlInt32 op\_Implicit(SqlInt16 x);

[VB] returnValue = SqlInt32.op\_Implicit(x)

[JScript] returnValue = x;

#### Description

Converts the supplied **System.Data.SqlTypes.SqlInt16** to **System.Data.SqlTypes.SqlInt32**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt32** structure whose **System.Data.SqlTypes.SqlInt32.Value** property equals the **System.Data.SqlTypes.SqlInt16.Value** property of the **System.Data.SqlTypes.SqlInt16** parameter. A **System.Data.SqlTypes.SqlInt16** structure.

op\_Inequality

[C#] public static SqlBoolean operator !=(SqlInt32 x, SqlInt32 y);

```

1 [C++] public: static SqlBoolean op_Inequality(SqlInt32 x, SqlInt32 y);
2 [VB]     returnValue = SqlInt32.op_Inequality(x, y)
3 [JScript]     returnValue = x != y;

```

#### 5 *Description*

6 Perform a logical comparison of the two **System.Data.SqlTypes.SqlInt32**  
7 parameters to determine if they are equal.

8 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
9 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or  
10 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either  
11 instance of **System.Data.SqlTypes.SqlInt32** is null, the  
12 **System.Data.SqlTypes.SqlBoolean.Value** of the  
13 **System.Data.SqlTypes.SqlBoolean** will be  
14 **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlInt32**  
15 structure. A **System.Data.SqlTypes.SqlInt32** structure.

16 **op\_LessThan**

```

17
18 [C#]     public static SqlBoolean operator
19 [C++] public: static SqlBoolean op_LessThan(SqlInt32 x, SqlInt32 y);
20 [VB]     returnValue = SqlInt32.op_LessThan(x, y)
21 [JScript]     returnValue = x < y;

```

#### 23 *Description*

24 Compares the two **System.Data.SqlTypes.SqlInt32** parameters to  
25 determine if the first is less than the second.

1 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 2 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the  
 3 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
 4 instance of **System.Data.SqlTypes.SqlInt32** is null, the  
 5 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 6 **System.Data.SqlTypes.SqlBoolean** will be  
 7 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt32**  
 8 structure. A **System.Data.SqlTypes.SqlInt32** structure.

9 op\_LessThanOrEqual

10  
 11 [C#] public static SqlBoolean operator <=(SqlInt32 x, SqlInt32 y);  
 12 [C++] public: static SqlBoolean op\_LessThanOrEqual(SqlInt32 x, SqlInt32 y);  
 13 [VB] returnValue = SqlInt32.op\_LessThanOrEqual(x, y)  
 14 [JScript] returnValue = x <= y;  
 15

## 16 *Description*

17 Compares the two **System.Data.SqlTypes.SqlInt32** parameters to  
 18 determine if the first is less than or equal to the second.

19 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 20 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal  
 21 to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If  
 22 either instance of **System.Data.SqlTypes.SqlInt32** is null, the  
 23 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 24 **System.Data.SqlTypes.SqlBoolean** will be  
 25

**System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt32** structure. A **System.Data.SqlTypes.SqlInt32** structure.

op\_Modulus

[C#] public static SqlInt32 operator %(SqlInt32 x, SqlInt32 y);

[C++] public: static SqlInt32 op\_Modulus(SqlInt32 x, SqlInt32 y);

[VB] returnValue = SqlInt32.op\_Modulus(x, y)

[JScript] returnValue = x % y;

#### *Description*

The modulus operator computes the remainder after dividing the first

**System.Data.SqlTypes.SqlInt32** parameter by the second.

*Return Value:* A **System.Data.SqlTypes.SqlInt32** structure whose

**System.Data.SqlTypes.SqlInt32.Value** contains the remainder. A

**System.Data.SqlTypes.SqlInt32** structure. A **System.Data.SqlTypes.SqlInt32**

structure.

op\_Multiply

[C#] public static SqlInt32 operator \*(SqlInt32 x, SqlInt32 y);

[C++] public: static SqlInt32 op\_Multiply(SqlInt32 x, SqlInt32 y);

[VB] returnValue = SqlInt32.op\_Multiply(x, y)

[JScript] returnValue = x \* y;

#### *Description*

1       The multiplication operator computes the product of the two  
2 **System.Data.SqlTypes.SqlInt32** parameters.

3 *Return Value:* A **System.Data.SqlTypes.SqlInt32** structure whose  
4 **System.Data.SqlTypes.SqlInt32.Value** contains the product of the two  
5 parameters. A **System.Data.SqlTypes.SqlInt32** structure. A  
6 **System.Data.SqlTypes.SqlInt32** structure.

7       op\_OnesComplement

8  
9 [C#]     public     static     SqlInt32     operator     ~(SqlInt32     x);  
10 [C++]    public:    static     SqlInt32     op\_OnesComplement(SqlInt32    x);  
11 [VB]       returnValue        =            SqlInt32.op\_OnesComplement(x)  
12 [JScript]        returnValue        =            ~x;

13  
14 *Description*

15       The ~ operator performs a bitwise one's complement operation on its  
16 **System.Data.SqlTypes.SqlInt32** operand. A **System.Data.SqlTypes.SqlInt32**  
17 structure.

18       op\_Subtraction

19  
20 [C#]     public     static     SqlInt32     operator     -(SqlInt32    x,   SqlInt32    y);  
21 [C++]    public:    static     SqlInt32     op\_Subtraction(SqlInt32    x,   SqlInt32    y);  
22 [VB]       returnValue        =            SqlInt32.op\_Subtraction(x,        y)  
23 [JScript]        returnValue        =            x        -        y;

24  
25 *Description*

1       The subtraction operator subtracts the second  
 2       **System.Data.SqlTypes.SqlInt32** parameter from the first.  
 3       *Return Value:* A **System.Data.SqlTypes.SqlInt32** structure whose  
 4       **System.Data.SqlTypes.SqlInt32.Value** property contains the results of the  
 5       subtraction. A **System.Data.SqlTypes.SqlInt32** structure. A  
 6       **System.Data.SqlTypes.SqlInt32** structure.

7       op\_UnaryNegation

9       [C#]       public       static       SqlInt32       operator       -(SqlInt32       x);  
 10       [C++]       public:       static       SqlInt32       op\_UnaryNegation(SqlInt32       x);  
 11       [VB]       returnValue       =       SqlInt32.op\_UnaryNegation(x)  
 12       [JScript]       returnValue       =       -x;

14       *Description*

15       The unary minus operator negates the  
 16       **System.Data.SqlTypes.SqlInt32.Value** of the **System.Data.SqlTypes.SqlInt32**  
 17       operand. A **System.Data.SqlTypes.SqlInt32** structure.

18       Parse

20       [C#]       public       static       SqlInt32       Parse(string       s);  
 21       [C++]       public:       static       SqlInt32       Parse(String\*       s);  
 22       [VB]       Public Shared Function Parse(ByVal s As String) As SqlInt32  
 23       [JScript]       public       static       function       Parse(s : String) : SqlInt32;

25       *Description*

```

1      [ .][ .]
2      Subtract
3
4  [C#]   public   static   SqlInt32   Subtract(SqlInt32   x,   SqlInt32   y);
5  [C++]   public:   static   SqlInt32   Subtract(SqlInt32   x,   SqlInt32   y);
6  [VB]   Public Shared Function Subtract(ByVal x As SqlInt32, ByVal y As
7  SqlInt32)                                     As                               SqlInt32
8  [JScript] public static function Subtract(x : SqlInt32, y : SqlInt32) : SqlInt32;
9

```

*Description*

```

11      [ .]
12      ToSqlBoolean
13
14  [C#]           public           SqlBoolean           ToSqlBoolean();
15  [C++]           public:           SqlBoolean           ToSqlBoolean();
16  [VB]   Public   Function   ToSqlBoolean()   As       SqlBoolean
17  [JScript]   public   function   ToSqlBoolean()   :       SqlBoolean;
18

```

*Description*

```

20      [ .]
21      ToSqlByte
22
23  [C#]           public           SqlByte           ToSqlByte();
24  [C++]           public:           SqlByte           ToSqlByte();
25  [VB]   Public   Function   ToSqlByte()   As       SqlByte

```

```
1 || [JScript]      public      function      ToSqlByte()      :      SqlByte;
```

2

3	<i>Description</i>
---	--------------------

4 [ . ]

5 ToSqlDecimal

6

```
7 || [C#] public SqlDecimal ToSqlDecimal();
```

```
8 || [C++]          public:          SqlDecimal          ToSqlDecimal();
```

9	[VB]	Public	Function	ToSqlDecimal()	As	SqlDecimal
---	------	--------	----------	----------------	----	------------

```
10 || [JScript] public function ToSqlDecimal() : SqlDecimal;
```

11

12 *Description*

13 [ .]

14 ToSqlDouble

15

```
16 || [C#] public SqlDouble ToSqlDouble();
```

```
17 || [C++] public: SqlDouble ToSqlDouble();
```

18	[VB]	Public	Function	ToSqlDouble()	As	SqlDouble
----	------	--------	----------	---------------	----	-----------

```
19 || [JScript] public function ToSqlDouble() : SqlDouble;
```

20

21 *Description*

22 [ .]

23 ToSqlInt16

24

```
25  [C#]      public      SqlInt16      ToSqlInt16());
```

```

1  [C++]          public:          SqlInt16          ToSqlInt16();
2  [VB]          Public          Function          ToSqlInt16()          As          SqlInt16
3  [JScript]      public          function          ToSqlInt16()          :          SqlInt16;

```

4  
5 *Description*

```

6      [ .]
7      ToSqlInt64

```

```

9  [C#]          public          SqlInt64          ToSqlInt64();
10 [C++]          public:          SqlInt64          ToSqlInt64();
11 [VB]          Public          Function          ToSqlInt64()          As          SqlInt64
12 [JScript]      public          function          ToSqlInt64()          :          SqlInt64;

```

13  
14 *Description*

```

15      [ .]
16      ToSqlMoney

```

```

18 [C#]          public          SqlMoney          ToSqlMoney();
19 [C++]          public:          SqlMoney          ToSqlMoney();
20 [VB]          Public          Function          ToSqlMoney()          As          SqlMoney
21 [JScript]      public          function          ToSqlMoney()          :          SqlMoney;

```

22  
23 *Description*

```

24      [ .]
25      ToSqlSingle

```

```

1
2 [C#]          public          SqlSingle          ToSqlSingle();
3 [C++]          public:          SqlSingle          ToSqlSingle();
4 [VB]      Public      Function      ToSqlSingle()      As      SqlSingle
5 [JScript]      public      function      ToSqlSingle()      :      SqlSingle;

```

6  
7 *Description*

```

8      [ .]
9      ToSqlString

```

```

10
11 [C#]          public          SqlString          ToSqlString();
12 [C++]          public:          SqlString          ToSqlString();
13 [VB]      Public      Function      ToSqlString()      As      SqlString
14 [JScript]      public      function      ToSqlString()      :      SqlString;

```

15  
16 *Description*

```

17      [ .]
18      ToString

```

```

19
20 [C#]          public          override          string          ToString();
21 [C++]          public:          String*          ToString();
22 [VB]      Overrides      Public      Function      ToString()      As      String
23 [JScript]      public      override      function      ToString()      :      String; Converts a
24 System.Data.SqlTypes.SqlInt32      structure      to      a      System.String .

```

25

*Description*

Converts a **System.Data.SqlTypes.SqlInt32** structure to a **System.String**.

Xor

[C#] public static SqlInt32 Xor(SqlInt32 x, SqlInt32 y);

[C++] public: static SqlInt32 Xor(SqlInt32 x, SqlInt32 y);

[VB] Public Shared Function Xor(ByVal x As SqlInt32, ByVal y As SqlInt32) As  
SqlInt32

[JScript] public static function Xor(x : SqlInt32, y : SqlInt32) : SqlInt32;

*Description*

[ .]

SqlInt64 structure (System.Data.SqlTypes)

Xor

*Description*

Represents a 64-bit signed integer to be stored in or retrieved from a  
database.

Xor

[C#] public static readonly SqlInt64 MaxValue;

[C++] public: static SqlInt64 MaxValue;

[VB] Public Shared ReadOnly MaxValue As SqlInt64

1 [JScript] public static var MaxValue : SqlInt64;

2

3 *Description*

4 A constant representing the largest possible value for a  
5 **System.Data.SqlTypes.SqlInt64** structure.

6 The value of this constant is 2<sup>63</sup> - 1.

7 Xor

8

9 [C#] public static readonly SqlInt64 MinValue;

10 [C++] public: static SqlInt64 MinValue;

11 [VB] Public Shared ReadOnly MinValue As SqlInt64

12 [JScript] public static var MinValue : SqlInt64;

13

14 *Description*

15 A constant representing the smallest possible value for a  
16 **System.Data.SqlTypes.SqlInt64** structure.

17 The value of this constant is -2<sup>63</sup>.

18 Xor

19

20 [C#] public static readonly SqlInt64 Null;

21 [C++] public: static SqlInt64 Null;

22 [VB] Public Shared ReadOnly Null As SqlInt64

23 [JScript] public static var Null : SqlInt64;

24

25 *Description*

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlInt64.Value** property of an instance of the **System.Data.SqlTypes.SqlInt64** structure.

**System.Data.SqlTypes.SqlInt64.Null** functions as a constant for the **System.Data.SqlTypes.SqlInt64** structure.

Xor

```
[C#]      public      static      readonly      SqlInt64      Zero;
[C++]      public:      static      SqlInt64      Zero;
[VB]      Public      Shared      ReadOnly      Zero      As      SqlInt64
[JScript]      public      static      var      Zero      :      SqlInt64;
```

*Description*

Represents a zero value that can be assigned to the **System.Data.SqlTypes.SqlInt64.Value** property of an instance of the **System.Data.SqlTypes.SqlInt64** structure.

The **System.Data.SqlTypes.SqlInt64.Zero** field is a constant for the **System.Data.SqlTypes.SqlInt64** structure.

SqlInt64

*Example Syntax:*

Xor

```
[C#]      public      SqlInt64(long      value);
[C++]      public:      SqlInt64(__int64      value);
[VB]      Public      Sub      New(ByVal      value      As      Long)
```

1 [JScript] public function SqlInt64(value : long);

2

3 *Description*

4 Initializes a new instance of the **System.Data.SqlTypes.SqlInt64** structure  
5 using the supplied long integer. A long integer.

6 IsNull

7 Xor

8

9 [C#] public bool IsNull {get;}

10 [C++] public: \_\_property bool get\_IsNull();

11 [VB] Public ReadOnly Property IsNull As Boolean

12 [JScript] public function get IsNull() : Boolean;

13

14 *Description*

15 Indicates whether or not **System.Data.SqlTypes.SqlInt64.Value** is null.

16 Value

17 Xor

18

19 [C#] public long Value {get;}

20 [C++] public: \_\_property \_\_int64 get\_Value();

21 [VB] Public ReadOnly Property Value As Long

22 [JScript] public function get Value() : long;

23

24 *Description*

25

Gets the value of this **System.Data.SqlTypes.SqlInt64** structure. This property is read-only.

### Add

[C#] public static SqlInt64 Add(SqlInt64 x, SqlInt64 y);

[C++] public: static SqlInt64 Add(SqlInt64 x, SqlInt64 y);

[VB] Public Shared Function Add(ByVal x As SqlInt64, ByVal y As SqlInt64) As SqlInt64

[JScript] public static function Add(x : SqlInt64, y : SqlInt64) : SqlInt64;

### Description

[ .]

### BitwiseAnd

[C#] public static SqlInt64 BitwiseAnd(SqlInt64 x, SqlInt64 y);

[C++] public: static SqlInt64 BitwiseAnd(SqlInt64 x, SqlInt64 y);

[VB] Public Shared Function BitwiseAnd(ByVal x As SqlInt64, ByVal y As SqlInt64) As SqlInt64

[JScript] public static function BitwiseAnd(x : SqlInt64, y : SqlInt64) : SqlInt64;

### Description

[ .]

### BitwiseOr

[C#] public static SqlInt64 BitwiseOr(SqlInt64 x, SqlInt64 y);

```

1 [C++] public: static SqlInt64 BitwiseOr(SqlInt64 x, SqlInt64 y);
2 [VB] Public Shared Function BitwiseOr(ByVal x As SqlInt64, ByVal y As
3 SqlInt64) As SqlInt64
4 [JScript] public static function BitwiseOr(x : SqlInt64, y : SqlInt64) : SqlInt64;

```

#### Description

[ .]

#### CompareTo

```

10 [C#] public int CompareTo(object value);
11 [C++] public: __sealed int CompareTo(Object* value);
12 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
13 Integer
14 [JScript] public function CompareTo(value : Object) : int;

```

#### Description

Compares this instance to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

#### Divide

```

23 [C#] public static SqlInt64 Divide(SqlInt64 x, SqlInt64 y);
24 [C++] public: static SqlInt64 Divide(SqlInt64 x, SqlInt64 y);
25 [VB] Public Shared Function Divide(ByVal x As SqlInt64, ByVal y As SqlInt64)

```

1 As SqlInt64  
 2 [JScript] public static function Divide(x : SqlInt64, y : SqlInt64) : SqlInt64;

3  
 4 *Description*

5 [ .]  
 6 Equals

7  
 8 [C#] public override bool Equals(object value);  
 9 [C++] public: bool Equals(Object\* value);  
 10 [VB] Overrides Public Function Equals(ByVal value As Object) As Boolean  
 11 [JScript] public override function Equals(value : Object) : Boolean;

12  
 13 *Description*  
 14 Compares the supplied object parameter to the  
 15 **System.Data.SqlTypes.SqlInt64.Value** property of the  
 16 **System.Data.SqlTypes.SqlInt64** object.

17 *Return Value:* **true** if object is an instance of **System.Data.SqlTypes.SqlInt64**  
 18 and the two are equal; otherwise **false** . The object to be compared.

19 Equals

20  
 21 [C#] public static new SqlBoolean Equals(SqlInt64 x, SqlInt64 y);  
 22 [C++] public: static SqlBoolean Equals(SqlInt64 x, SqlInt64 y);  
 23 [VB] Shadows Public Shared Function Equals(ByVal x As SqlInt64, ByVal y As  
 24 SqlInt64) As SqlBoolean  
 25 [JScript] public static hide function Equals(x : SqlInt64, y : SqlInt64) :

1 SqlBoolean;

3 *Description*

4 [ . ]

5 GetHashCode

7 [C#] public override int GetHashCode();

8 [C++] public: int GetHashCode();

9 [VB] Overrides Public Function GetHashCode() As Integer

10 [JScript] public override function GetHashCode() : int;

12 *Description*

13 Returns the hash code for this instance.

14 *Return Value:* A 32-bit signed integer hash code.

15 GreaterThan

17 [C#] public static SqlBoolean GreaterThan(SqlInt64 x, SqlInt64 y);

18 [C++] public: static SqlBoolean GreaterThan(SqlInt64 x, SqlInt64 y);

19 [VB] Public Shared Function GreaterThan(ByVal x As SqlInt64, ByVal y As  
20 SqlInt64) As SqlBoolean

21 [JScript] public static function GreaterThan(x : SqlInt64, y : SqlInt64) :  
22 SqlBoolean;

24 *Description*

25 [ . ]

## GreaterThanOrEqualTo

```
[C#] public static SqlBoolean GreaterThanOrEqualTo(SqlInt64 x, SqlInt64 y);  
[C++] public: static SqlBoolean GreaterThanOrEqualTo(SqlInt64 x, SqlInt64 y);  
[VB] Public Shared Function GreaterThanOrEqualTo(ByVal x As SqlInt64, ByVal y  
As SqlInt64) As SqlBoolean  
[JScript] public static function GreaterThanOrEqualTo(x : SqlInt64, y : SqlInt64) :  
SqlBoolean;
```

### Description

[ .]

LessThan

```
[C#] public static SqlBoolean LessThan(SqlInt64 x, SqlInt64 y);  
[C++] public: static SqlBoolean LessThan(SqlInt64 x, SqlInt64 y);  
[VB] Public Shared Function LessThan(ByVal x As SqlInt64, ByVal y As  
SqlInt64) As SqlBoolean  
[JScript] public static function LessThan(x : SqlInt64, y : SqlInt64) : SqlBoolean;
```

### Description

[ .]

LessThanOrEqualTo

```
[C#] public static SqlBoolean LessThanOrEqualTo(SqlInt64 x, SqlInt64 y);  
[C++] public: static SqlBoolean LessThanOrEqualTo(SqlInt64 x, SqlInt64 y);
```

1 [VB] Public Shared Function LessThanOrEqual(ByVal x As SqlInt64, ByVal y As  
2 SqlInt64) As SqlBoolean

3 [JScript] public static function LessThanOrEqual(x : SqlInt64, y : SqlInt64) :  
4 SqlBoolean;

5  
6 *Description*

7 [ .]

8 Mod

9  
10 [C#] public static SqlInt64 Mod(SqlInt64 x, SqlInt64 y);

11 [C++] public: static SqlInt64 Mod(SqlInt64 x, SqlInt64 y);

12 [VB] Public Shared Function Mod(ByVal x As SqlInt64, ByVal y As SqlInt64) As  
13 SqlInt64

14 [JScript] public static function Mod(x : SqlInt64, y : SqlInt64) : SqlInt64;

15  
16 *Description*

17 [ .]

18 Multiply

19  
20 [C#] public static SqlInt64 Multiply(SqlInt64 x, SqlInt64 y);

21 [C++] public: static SqlInt64 Multiply(SqlInt64 x, SqlInt64 y);

22 [VB] Public Shared Function Multiply(ByVal x As SqlInt64, ByVal y As  
23 SqlInt64) As SqlInt64

24 [JScript] public static function Multiply(x : SqlInt64, y : SqlInt64) : SqlInt64;

25

*Description*

[ .]

NotEquals

[C#] public static SqlBoolean NotEquals(SqlInt64 x, SqlInt64 y);

[C++] public: static SqlBoolean NotEquals(SqlInt64 x, SqlInt64 y);

[VB] Public Shared Function NotEquals(ByVal x As SqlInt64, ByVal y As SqlInt64) As SqlBoolean

[JScript] public static function NotEquals(x : SqlInt64, y : SqlInt64) : SqlBoolean;

*Description*

[ .]

OnesComplement

[C#] public static SqlInt64 OnesComplement(SqlInt64 x);

[C++] public: static SqlInt64 OnesComplement(SqlInt64 x);

[VB] Public Shared Function OnesComplement(ByVal x As SqlInt64) As SqlInt64

[JScript] public static function OnesComplement(x : SqlInt64) : SqlInt64;

*Description*

[ .]

op\_Addition

```

1
2 [C#] public static SqlInt64 operator +(SqlInt64 x, SqlInt64 y);
3 [C++] public: static SqlInt64 op_Addition(SqlInt64 x, SqlInt64 y);
4 [VB]     returnValue          =          SqlInt64.op_Addition(x,          y)
5 [JScript]     returnValue          =          x          +          y;
6

```

#### *Description*

The addition operator computes the sum of the two **System.Data.SqlTypes.SqlInt64** parameters.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** is equal to the sum of the two **System.Data.SqlTypes.SqlInt64** parameters. A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

#### **op\_BitwiseAnd**

```

16 [C#] public static SqlInt64 operator &(amp;SqlInt64 x, SqlInt64 y);
17 [C++] public: static SqlInt64 op_BitwiseAnd(SqlInt64 x, SqlInt64 y);
18 [VB]     returnValue          =          SqlInt64.op_BitwiseAnd(x,          y)
19 [JScript]     returnValue          =          x          &          y;
20

```

#### *Description*

Computes the bitwise AND of its **System.Data.SqlTypes.SqlInt64** operands. A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

#### **op\_BitwiseOr**

```

1
2 [C#] public static SqlInt64 operator |(SqlInt64 x, SqlInt64 y);
3 [C++] public: static SqlInt64 op_BitwiseOr(SqlInt64 x, SqlInt64 y);
4 [VB]     returnValue      =      SqlInt64.op_BitwiseOr(x,      y)
5 [JScript]     returnValue      =      x      |      y;

```

#### *Description*

Computes the bitwise OR of its two **System.Data.SqlTypes.SqlInt64** operands. A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

op\_Division

```

13 [C#] public static SqlInt64 operator /(SqlInt64 x, SqlInt64 y);
14 [C++] public: static SqlInt64 op_Division(SqlInt64 x, SqlInt64 y);
15 [VB]     returnValue      =      SqlInt64.op_Division(x,      y)
16 [JScript]     returnValue      =      x      /      y;

```

#### *Description*

The division operator divides the first **System.Data.SqlTypes.SqlInt64** parameter by the second.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** property contains the results of the division operation. A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

op\_Equality

```

1
2 [C#] public static SqlBoolean operator ==(SqlInt64 x, SqlInt64 y);
3 [C++] public: static SqlBoolean op_Equality(SqlInt64 x, SqlInt64 y);
4 [VB]     returnValue      =      SqlInt64.op_Equality(x,      y)
5 [JScript]     returnValue      =      x      ==      y;
6

```

### 7 *Description*

8 Performs a logical comparison of the two **System.Data.SqlTypes.SqlInt64**  
9 parameters to determine if they are equal.

10 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
11 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or  
12 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If  
13 either instance of **System.Data.SqlTypes.SqlInt64** is null, the  
14 **System.Data.SqlTypes.SqlBoolean.Value** of the  
15 **System.Data.SqlTypes.SqlBoolean** will be  
16 **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlInt64**  
17 structure. A **System.Data.SqlTypes.SqlInt64** structure.

18 op\_ExclusiveOr

```

19
20 [C#] public static SqlInt64 operator ^(SqlInt64 x, SqlInt64 y);
21 [C++] public: static SqlInt64 op_ExclusiveOr(SqlInt64 x, SqlInt64 y);
22 [VB]     returnValue      =      SqlInt64.op_ExclusiveOr(x,      y)
23 [JScript]     returnValue      =      x      ^      y;
24

```

### 25 *Description*

Performs a bitwise exclusive-OR operation on the supplied parameters. A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

op\_Explicit

```
[C#]    public    static    explicit    operator    SqlInt64(SqlBoolean    x);
[C++]    public:    static    SqlInt64    op_Explicit(SqlBoolean    x);
[VB]        returnValue    =    SqlInt64.op_Explicit(x)
[JScript]        returnValue    =    SqlInt64(x);
```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlBit** parameter to **System.Data.SqlTypes.SqlInt64**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** property is equal to the **System.Data.SqlTypes.SqlBit.ByteValue** of the **System.Data.SqlTypes.SqlBit** parameter. The **System.Data.SqlTypes.SqlBit** structure to be converted.

op\_Explicit

```
[C#]    public    static    explicit    operator    SqlInt64(SqlDecimal    x);
[C++]    public:    static    SqlInt64    op_Explicit(SqlDecimal    x);
[VB]        returnValue    =    SqlInt64.op_Explicit(x)
[JScript]        returnValue    =    SqlInt64(x);
```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlDecimal** parameter to **System.Data.SqlTypes.SqlInt64**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** is equal to the integer portion of the **System.Data.SqlTypes.SqlDecimal** parameter. The **System.Data.SqlTypes.SqlDecimal** structure to be converted.

op\_Explicit

```
[C#]      public      static      explicit      operator      SqlInt64(SqlDouble      x);
[C++]      public:      static      SqlInt64      op_Explicit(SqlDouble      x);
[VB]      returnValue      =      SqlInt64.op_Explicit(x)
[JScript]      returnValue      =      SqlInt64(x);
```

#### Description

Converts the supplied **System.Data.SqlTypes.SqlDouble** structure to **System.Data.SqlTypes.SqlInt64**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** property equals the integer portion of the **System.Data.SqlTypes.SqlDouble** parameter. The **System.Data.SqlTypes.SqlDouble** structure to be converted.

op\_Explicit

```
[C#]      public      static      explicit      operator      long(SqlInt64      x);
[C++]      public:      static      __int64      op_Explicit();
[VB]      returnValue      =      SqlInt64.op_Explicit(x)
```

1 [JScript]                      returnValue                      =                      Int64(x);

2

3 *Description*

4            Converts the **System.Data.SqlTypes.SqlInt64** parameter to long.

5 *Return Value:*    A new long value equal to the

6 **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64** .

7 A **System.Data.SqlTypes.SqlInt64** structure.

8            op\_Explicit

9

10 [C#]        public        static        explicit        operator        SqlInt64(SqlMoney        x);

11 [C++]        public:        static        SqlInt64        op\_Explicit(SqlMoney        x);

12 [VB]                returnValue                =                SqlInt64.op\_Explicit(x)

13 [JScript]                returnValue                =                SqlInt64(x);

14

15 *Description*

16            Converts the supplied **System.Data.SqlTypes.SqlMoney** parameter to

17 **System.Data.SqlTypes.SqlInt64** . The **System.Data.SqlTypes.SqlMoney**

18 structure to be converted.

19            op\_Explicit

20

21 [C#]        public        static        explicit        operator        SqlInt64(SqlSingle        x);

22 [C++]        public:        static        SqlInt64        op\_Explicit(SqlSingle        x);

23 [VB]                returnValue                =                SqlInt64.op\_Explicit(x)

24 [JScript]                returnValue                =                SqlInt64(x);

25

*Description*

Converts the supplied **System.Data.SqlTypes.SqlSingle** parameter to **SqlInt64**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** property contains the integer portion of the **System.Data.SqlTypes.SqlSingle** parameter. The **System.Data.SqlTypes.SqlSingle** structure to be converted.

**op\_Explicit**

```
[C#]    public static explicit operator SqlInt64(SqlString x);
[C++]    public: static SqlInt64 op_Explicit(SqlString x);
[VB]    returnValue = SqlInt64.op_Explicit(x)
[JScript]    returnValue = SqlInt64(x);
```

*Description*

Converts the supplied **System.Data.SqlTypes.SqlString** parameter to **System.Data.SqlTypes.SqlInt64**.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** whose **System.Data.SqlTypes.SqlInt64.Value** is equal to the value represented by the **System.Data.SqlTypes.SqlString** parameter. The **System.Data.SqlTypes.SqlString** object to be converted.

**op\_GreaterThan**

```
[C#]    public static SqlBoolean operator >(SqlInt64 x, SqlInt64 y);
```

```

1 [C++] public: static SqlBoolean op_GreaterThan(SqlInt64 x, SqlInt64 y);
2 [VB]      returnValue      =      SqlInt64.op_GreaterThan(x,      y)
3 [JScript]      returnValue      =      x      >      y;

```

#### *Description*

Performs a logical comparison of the two **System.Data.SqlTypes.SqlInt64** parameters to determine if the first is greater than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlInt64** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

**op\_GreaterThanOrEqual**

```

18 [C#] public static SqlBoolean operator >=(SqlInt64 x, SqlInt64 y);
19 [C++] public: static SqlBoolean op_GreaterThanOrEqual(SqlInt64 x, SqlInt64 y);
20 [VB]      returnValue      =      SqlInt64.op_GreaterThanOrEqual(x,      y)
21 [JScript]      returnValue      =      x      >=      y;

```

#### *Description*

Performs a logical comparison of the two **System.Data.SqlTypes.SqlInt64** parameters to determine if the first is greater than or equal to the second.

1 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 2 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greaater than or  
 3 equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**  
 4 . If either instance of **System.Data.SqlTypes.SqlInt64** is null, the  
 5 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 6 **System.Data.SqlTypes.SqlBoolean** will be  
 7 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt64**  
 8 structure. A **System.Data.SqlTypes.SqlInt64** structure.

9 **op\_Implicit**  
 10  
 11 [C#] public static implicit operator SqlInt64(long x);  
 12 [C++] public: static SqlInt64 op\_Implicit(\_\_int64 x);  
 13 [VB] returnValue = SqlInt64.op\_Implicit(x)  
 14 [JScript] returnValue = x;

# *Description*

17 Converts the long parameter to **System.Data.SqlTypes.SqlInt64** .  
 18 *Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose  
 19 **System.Data.SqlTypes.SqlInt64.Value** equals the value of the long parameter. A  
 20 long integer value.

21 **op\_Implicit**  
 22  
 23 [C#] public static implicit operator SqlInt64(SqlByte x);  
 24 [C++] public: static SqlInt64 op\_Implicit(SqlByte x);  
 25 [VB] returnValue = SqlInt64.op\_Implicit(x)

1 [JScript]                      returnValue                      =                      x;

2

3 *Description*

4        Converts the supplied **System.Data.SqlTypes.SqlByte** parameter to  
5 **System.Data.SqlTypes.SqlInt64**

6 *Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose  
7 **System.Data.SqlTypes.SqlInt64.Value** property equals the  
8 **System.Data.SqlTypes.SqlByte.Value** property of the  
9 **System.Data.SqlTypes.SqlByte** parameter. The **System.Data.SqlTypes.SqlByte**  
10 structure to be converted.

11        op\_Implicit

12

13 [C#]        public        static        implicit        operator        SqlInt64(SqlInt16        x);

14 [C++]        public:        static        SqlInt64        op\_Implicit(SqlInt16        x);

15 [VB]                returnValue                =                SqlInt64.op\_Implicit(x)

16 [JScript]                returnValue                =                x;

17

18 *Description*

19        Converts the supplied **System.Data.SqlTypes.SqlInt16** parameter to  
20 **System.Data.SqlTypes.SqlInt64**

21 *Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose  
22 **System.Data.SqlTypes.SqlInt64.Value** property equals the  
23 **System.Data.SqlTypes.SqlInt16.Value** property of the  
24 **System.Data.SqlTypes.SqlInt16** parameter. The  
25 **System.Data.SqlTypes.SqlInt16** structure to be converted.

```

1      op_Implicit
2
3  [C#]    public    static    implicit    operator    SqlInt64(SqlInt32    x);
4  [C++]   public:   static    SqlInt64    op_Implicit(SqlInt32    x);
5  [VB]           returnValue          =          SqlInt64.op_Implicit(x)
6  [JScript]           returnValue          =          x;

```

### Description

Converts the supplied **System.Data.SqlTypes.SqlInt32** parameter to **System.Data.SqlTypes.SqlInt64**.

**Return Value:** A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** property equals the **System.Data.SqlTypes.SqlInt32.Value** property of the **System.Data.SqlTypes.SqlInt32** parameter. The **System.Data.SqlTypes.SqlInt32** structure to be converted.

```

16      op_Inequality
17
18  [C#]    public    static    SqlBoolean    operator    !=(SqlInt64    x,    SqlInt64    y);
19  [C++]   public:   static    SqlBoolean    op_Inequality(SqlInt64    x,    SqlInt64    y);
20  [VB]           returnValue          =          SqlInt64.op_Inequality(x,          y)
21  [JScript]           returnValue          =          x          !=          y;

```

### Description

Performs a logical comparison on the two **SqlInt64** parameters to determine if they are equal.

1 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 2 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or  
 3 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either  
 4 instance of **System.Data.SqlTypes.SqlInt64** is null, the  
 5 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 6 **System.Data.SqlTypes.SqlBoolean** will be  
 7 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt64**  
 8 structure. A **System.Data.SqlTypes.SqlInt64** structure.

9       op\_LessThan  
 10  
 11 [C#]       public       static       SqlBoolean       operator  
 12 [C++]   public: static   SqlBoolean   op\_LessThan(SqlInt64 x, SqlInt64 y);  
 13 [VB]       returnValue       =       SqlInt64.op\_LessThan(x,       y)  
 14 [JScript]       returnValue       =       x       <       y;  
 15

16 *Description*  
 17       Performs a logical comparison on the two **System.Data.SqlTypes.SqlInt64**  
 18 parameters to determine if the first is less than the second.

19 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
 20 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the  
 21 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
 22 instance of **System.Data.SqlTypes.SqlInt64** is null, the  
 23 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 24 **System.Data.SqlTypes.SqlBoolean** will be  
 25

**System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

**op\_LessThanOrEqual**

[C#] public static SqlBoolean operator <=(SqlInt64 x, SqlInt64 y);  
 [C++] public: static SqlBoolean op\_LessThanOrEqual(SqlInt64 x, SqlInt64 y);  
 [VB] returnValue = SqlInt64.op\_LessThanOrEqual(x, y)  
 [JScript] returnValue = x <= y;

### *Description*

Performs a logical comparison on the two **System.Data.SqlTypes.SqlInt64** parameters to determine if the first is less than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **System.Data.SqlTypes.SqlInt64** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

**op\_Modulus**

[C#] public static SqlInt64 operator %(SqlInt64 x, SqlInt64 y);  
 [C++] public: static SqlInt64 op\_Modulus(SqlInt64 x, SqlInt64 y);  
 [VB] returnValue = SqlInt64.op\_Modulus(x, y)

1 [JScript]           returnValue           =           x           %           y;  
2

3 *Description*

4       The modulus operator computes the remainder after dividing the first  
5 **System.Data.SqlTypes.SqlInt64**       parameter       by       the       second.  
6 *Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose  
7 **System.Data.SqlTypes.SqlInt64.Value** property contains the remainder. A  
8 **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64**  
9 structure.

10       op\_Multiply

11  
12 [C#]   public   static   SqlInt64   operator   \*(SqlInt64   x,   SqlInt64   y);  
13 [C++]   public:   static   SqlInt64   op\_Multiply(SqlInt64   x,   SqlInt64   y);  
14 [VB]       returnValue           =           SqlInt64.op\_Multiply(x,           y)  
15 [JScript]       returnValue           =           x           \*           y;  
16

17 *Description*

18       The multiplication operator computes the product of the two  
19 **System.Data.SqlTypes.SqlInt64**                               parameters.  
20 *Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose  
21 **System.Data.SqlTypes.SqlInt64.Value** is equal to the product of the two  
22 **System.Data.SqlTypes.SqlInt64** parameters. A **System.Data.SqlTypes.SqlInt64**  
23 structure. A **System.Data.SqlTypes.SqlInt64** structure.

24       op\_OnesComplement

25

```

1
2 [C#]      public      static      SqlInt64      operator      ~(SqlInt64      x);
3 [C++]     public:     static      SqlInt64      op_OnesComplement(SqlInt64      x);
4 [VB]      returnValue      =      SqlInt64.op_OnesComplement(x)
5 [JScript]      returnValue      =      ~x;

```

### Description

The ~ operator performs a bitwise one's complement operation on its **System.Data.SqlTypes.SqlInt64** operand.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** is equal to the ones compliment of the **System.Data.SqlTypes.SqlInt64** parameter. A **System.Data.SqlTypes.SqlInt64** structure.

### op\_Subtraction

```

16 [C#]      public      static      SqlInt64      operator      -(SqlInt64      x,      SqlInt64      y);
17 [C++]     public:     static      SqlInt64      op_Subtraction(SqlInt64      x,      SqlInt64      y);
18 [VB]      returnValue      =      SqlInt64.op_Subtraction(x,      y)
19 [JScript]      returnValue      =      x      -      y;

```

### Description

The subtraction operator subtracts the second **System.Data.SqlTypes.SqlInt64** parameter from the first.

*Return Value:* A new **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** property equals the results of the

subtraction operation. A **System.Data.SqlTypes.SqlInt64** structure. A **System.Data.SqlTypes.SqlInt64** structure.

op\_UnaryNegation

[C#] public static SqlInt64 operator -(SqlInt64 x);

[C++] public: static SqlInt64 op\_UnaryNegation(SqlInt64 x);

[VB] returnValue = SqlInt64.op\_UnaryNegation(x)

[JScript] returnValue = -x;

### *Description*

The unary minus operator negates the **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64** operand.

*Return Value:* A **System.Data.SqlTypes.SqlInt64** structure whose **System.Data.SqlTypes.SqlInt64.Value** is equal to the negated **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64** parameter. A **System.Data.SqlTypes.SqlInt64** structure.

Parse

[C#] public static SqlInt64 Parse(string s);

[C++] public: static SqlInt64 Parse(String\* s);

[VB] Public Shared Function Parse(ByVal s As String) As SqlInt64

[JScript] public static function Parse(s : String) : SqlInt64;

### *Description*

```

1      [ .][ .]
2      Subtract
3
4      [C#]   public   static   SqlInt64   Subtract(SqlInt64   x,   SqlInt64   y);
5      [C++]  public:   static   SqlInt64   Subtract(SqlInt64   x,   SqlInt64   y);
6      [VB]   Public Shared Function Subtract(ByVal x As SqlInt64, ByVal y As
7      SqlInt64)                                     As                               SqlInt64
8      [JScript] public static function Subtract(x : SqlInt64, y : SqlInt64) : SqlInt64;
9

```

*Description*

```

11     [ .]
12     ToSqlBoolean
13
14     [C#]           public           SqlBoolean           ToSqlBoolean();
15     [C++]          public:          SqlBoolean           ToSqlBoolean();
16     [VB]           Public           Function           ToSqlBoolean() As           SqlBoolean
17     [JScript]      public           function           ToSqlBoolean() :           SqlBoolean;
18

```

*Description*

```

20     [ .]
21     ToSqlByte
22
23     [C#]           public           SqlByte           ToSqlByte();
24     [C++]          public:          SqlByte           ToSqlByte();
25     [VB]           Public           Function           ToSqlByte() As           SqlByte

```

1 [JScript] public function ToSqlByte() : SqlByte;

2

3 *Description*

4 [ .]

5 ToSqlDecimal

6

7 [C#] public SqlDecimal ToSqlDecimal();

8 [C++] public: SqlDecimal ToSqlDecimal();

9 [VB] Public Function ToSqlDecimal() As SqlDecimal

10 [JScript] public function ToSqlDecimal() : SqlDecimal;

11

12 *Description*

13 [ .]

14 ToSqlDouble

15

16 [C#] public SqlDouble ToSqlDouble();

17 [C++] public: SqlDouble ToSqlDouble();

18 [VB] Public Function ToSqlDouble() As SqlDouble

19 [JScript] public function ToSqlDouble() : SqlDouble;

20

21 *Description*

22 [ .]

23 ToSqlInt16

24

25 [C#] public SqlInt16 ToSqlInt16();

1 [C++] public: SqlInt16 ToSqlInt16();  
 2 [VB] Public Function ToSqlInt16() As SqlInt16  
 3 [JScript] public function ToSqlInt16() : SqlInt16;

4  
 5 *Description*

6 [ .]  
 7 ToSqlInt32

8  
 9 [C#] public SqlInt32 ToSqlInt32();  
 10 [C++] public: SqlInt32 ToSqlInt32();  
 11 [VB] Public Function ToSqlInt32() As SqlInt32  
 12 [JScript] public function ToSqlInt32() : SqlInt32;

13  
 14 *Description*

15 [ .]  
 16 ToSqlMoney

17  
 18 [C#] public SqlMoney ToSqlMoney();  
 19 [C++] public: SqlMoney ToSqlMoney();  
 20 [VB] Public Function ToSqlMoney() As SqlMoney  
 21 [JScript] public function ToSqlMoney() : SqlMoney;

22  
 23 *Description*

24 [ .]  
 25 ToSqlSingle

```

1
2 [C#]          public          SqlSingle          ToSqlSingle();
3 [C++]         public:         SqlSingle          ToSqlSingle();
4 [VB]         Public          Function          ToSqlSingle() As          SqlSingle
5 [JScript]     public          function          ToSqlSingle() :          SqlSingle;

```

6  
7 *Description*

```

8     [ .]
9     ToSqlString

```

```

10
11 [C#]          public          SqlString          ToSqlString();
12 [C++]         public:         SqlString          ToSqlString();
13 [VB]         Public          Function          ToSqlString() As          SqlString
14 [JScript]     public          function          ToSqlString() :          SqlString;

```

15  
16 *Description*

```

17     [ .]
18     ToString

```

```

19
20 [C#]          public          override          string          ToString();
21 [C++]         public:         String*          ToString();
22 [VB]         Overrides      Public          Function          ToString() As          String
23 [JScript]     public          override          function          ToString() : String; Converts a
24 System.Data.SqlTypes.SqlInt64      structure      to      System.String .

```

25

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Converts this instance of **System.Data.SqlTypes.SqlInt64** to **System.String** .

Xor

```
[C#]    public static SqlInt64 Xor(SqlInt64 x, SqlInt64 y);
[C++]   public: static SqlInt64 Xor(SqlInt64 x, SqlInt64 y);
[VB]    Public Shared Function Xor(ByVal x As SqlInt64, ByVal y As SqlInt64) As
SqlInt64
[JScrip] public static function Xor(x : SqlInt64, y : SqlInt64) : SqlInt64;
```

*Description*

[ .]  
SqlMoney structure (System.Data.SqlTypes)  
Xor

*Description*

Represents a currency value ranging from -2 (or - 922,337,203,685,477.5808) to 2 -1 (or +922,337,203,685,477.5807) with an accuracy to a ten-thousandth of currency unit to be stored in or retrieved from a database.

The actual value of the **System.Data.SqlTypes.SqlMoney** object is stored in **System.Data.SqlTypes.SqlMoney.Value** .

```

1      Xor
2
3  [C#]      public      static      readonly      SqlMoney      MaxValue;
4  [C++]      public:      static      SqlMoney      MaxValue;
5  [VB]      Public      Shared      ReadOnly      MaxValue      As      SqlMoney
6  [JScript]      public      static      var      MaxValue      :      SqlMoney;

```

#### 8 *Description*

9 Represents the maximum value that can be assigned to the  
10 **System.Data.SqlTypes.SqlMoney.Value** property of an instance of the  
11 **System.Data.SqlTypes.SqlMoney** class.

12 The value of this constant is 922,337,203,685,475.5807 Represents the  
13 maximum value that can be assigned to the  
14 **System.Data.SqlTypes.SqlMoney.Value** property of an instance of the  
15 **System.Data.SqlTypes.SqlMoney** class.

```

16      Xor
17
18  [C#]      public      static      readonly      SqlMoney      MinValue;
19  [C++]      public:      static      SqlMoney      MinValue;
20  [VB]      Public      Shared      ReadOnly      MinValue      As      SqlMoney
21  [JScript]      public      static      var      MinValue      :      SqlMoney;

```

#### 23 *Description*

24

25

Represents the minimum value that can be assigned to **System.Data.SqlTypes.SqlMoney.Value** property of an instance of the **System.Data.SqlTypes.SqlMoney** class.

The value of this constant is -922,337,203,685,477.5808 Represents the minimum value that can be assigned to **System.Data.SqlTypes.SqlMoney.Value** property of an instance of the **System.Data.SqlTypes.SqlMoney** class.

Xor

[C#]	public	static	readonly	SqlMoney	Null;
[C++]	public:	static		SqlMoney	Null;
[VB]	Public	Shared	ReadOnly	Null	As SqlMoney
[JScript]	public	static	var	Null	: SqlMoney;

#### Description

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlMoney.Value** property of an instance of the **System.Data.SqlTypes.SqlMoney** class.

**System.Data.SqlTypes.SqlMoney.Null** functions as a constant for the **System.Data.SqlTypes.SqlMoney** class.

Xor

[C#]	public	static	readonly	SqlMoney	Zero;
[C++]	public:	static		SqlMoney	Zero;
[VB]	Public	Shared	ReadOnly	Zero	As SqlMoney
[JScript]	public	static	var	Zero	: SqlMoney;

## Description

Represents the zero value that can be assigned to the **System.Data.SqlTypes.SqlMoney.Value** property of an instance of the **System.Data.SqlTypes.SqlMoney** class.

**System.Data.SqlTypes.SqlMoney.Zero** functions as a constant for the **System.Data.SqlTypes.SqlMoney** class.

SqlMoney

*Example Syntax:*

Xor

[C#]                    public                    SqlMoney(decimal                    value);

[C++]                    public:                    SqlMoney(Decimal                    value);

[VB]            Public            Sub            New(ByVal            value            As            Decimal)

[JScript]            public            function            SqlMoney(value            :            Decimal);

## Description

Initializes a new instance of the **System.Data.SqlTypes.SqlMoney** class with the value given. The monetary value to initialize.

SqlMoney

*Example Syntax:*

Xor

[C#]                    public                    SqlMoney(double                    value);

[C++]                    public:                    SqlMoney(double                    value);

1 [VB] Public Sub New(ByVal value As Double)  
2 [JScript] public function SqlMoney(value : double);  
3

4 *Description*  
5       Initializes a new instance of the **System.Data.SqlTypes.SqlMoney** class  
6 with the value given. The monetary value to initialize.

7       SqlMoney  
8       *Example Syntax:*  
9       Xor

10  
11 [C#] public SqlMoney(int value);  
12 [C++] public: SqlMoney(int value);  
13 [VB] Public Sub New(ByVal value As Integer)  
14 [JScript] public function SqlMoney(value : int);  
15

16 *Description*  
17       Initializes a new instance of the **System.Data.SqlTypes.SqlMoney** class  
18 with the value given. The monetary value to initialize.

19       SqlMoney  
20       *Example Syntax:*  
21       Xor

22  
23 [C#] public SqlMoney(long value);  
24 [C++] public: SqlMoney(\_\_int64 value);  
25 [VB] Public Sub New(ByVal value As Long)

1 [JScript]        public        function        SqlMoney(value        :        long);

2

3 *Description*

4        Initializes a new instance of the **System.Data.SqlTypes.SqlMoney** class  
5 with the value given. The monetary value to initialize.

6        IsNull

7        Xor

8

9 [C#]                public                bool                IsNull                {get;}

10 [C++]              public:              \_\_property              bool              get\_IsNull();

11 [VB]        Public        ReadOnly        Property        IsNull        As        Boolean

12 [JScript]        public        function        get        IsNull()        :        Boolean;

13

14 *Description*

15        Returns        a        value        indicating        whether        the  
16 **System.Data.SqlTypes.SqlMoney.Value** property is assigned to null.

17        Value

18        Xor

19

20 [C#]                public                decimal                Value                {get;}

21 [C++]              public:              \_\_property              Decimal              get\_Value();

22 [VB]        Public        ReadOnly        Property        Value        As        Decimal

23 [JScript]        public        function        get        Value()        :        Decimal;

24

25 *Description*

Gets the monetary value of an instance of the **System.Data.SqlTypes.SqlMoney** structure. This property is read-only.

#### Add

```
[C#] public static SqlMoney Add(SqlMoney x, SqlMoney y);
[C++] public: static SqlMoney Add(SqlMoney x, SqlMoney y);
[VB] Public Shared Function Add(ByVal x As SqlMoney, ByVal y As SqlMoney)
    As SqlMoney
[JavaScript] public static function Add(x : SqlMoney, y : SqlMoney) : SqlMoney;
```

#### Description

[ . ]

#### CompareTo

```
[C#] public int CompareTo(object value);
[C++] public: __sealed int CompareTo(Object* value);
[VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
    Integer
[JavaScript] public function CompareTo(value : Object) : int;
```

#### Description

Compares this instance to the supplied object and returns an indication of their relative values.

**Return Value:** A signed number indicating the relative values of the instance and the object. The object to be compared.

## Divide

```
[C#] public static SqlMoney Divide(SqlMoney x, SqlMoney y);  
[C++] public: static SqlMoney Divide(SqlMoney x, SqlMoney y);  
[VB] Public Shared Function Divide(ByVal x As SqlMoney, ByVal y As  
SqlMoney) As SqlMoney  
[JScript] public static function Divide(x : SqlMoney, y : SqlMoney) : SqlMoney;
```

### *Description*

[ .]

### Equals

```
[C#] public override bool Equals(object value);  
[C++] public: bool Equals(Object* value);  
[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean  
[JScript] public override function Equals(value : Object) : Boolean;
```

### *Description*

Compares the supplied object parameter to the **System.Data.SqlTypes.SqlMoney.Value** property of the **System.Data.SqlTypes.SqlMoney** object.

*Return Value:* Equals will return **true** if the object is an instance of **System.Data.SqlTypes.SqlMoney** and the two are equal; otherwise **false** . The object to be compared.

### Equals

```

1
2 [C#] public static new SqlBoolean Equals(SqlMoney x, SqlMoney y);
3 [C++] public: static SqlBoolean Equals(SqlMoney x, SqlMoney y);
4 [VB] Shadows Public Shared Function Equals(ByVal x As SqlMoney, ByVal y As
5 SqlMoney) As SqlBoolean
6 [JScript] public static hide function Equals(x : SqlMoney, y : SqlMoney) :
7 SqlBoolean;
8

```

*Description*

[ .]

GetHashCode

```

12
13 [C#] public override int GetHashCode();
14 [C++] public: int GetHashCode();
15 [VB] Overrides Public Function GetHashCode() As Integer
16 [JScript] public override function GetHashCode() : int;
17

```

*Description*

Gets the hash code for this instance.

*Return Value:* A 32-bit signed integer hash code.

GreaterThan

```

22
23 [C#] public static SqlBoolean GreaterThan(SqlMoney x, SqlMoney y);
24 [C++] public: static SqlBoolean GreaterThan(SqlMoney x, SqlMoney y);
25 [VB] Public Shared Function GreaterThan(ByVal x As SqlMoney, ByVal y As

```

```

1  SqlMoney)                                As                                SqlBoolean
2  [JScript] public static function GreaterThan(x : SqlMoney, y : SqlMoney) :
3  SqlBoolean;
    
```

*Description*

[ .]

GreaterThanOrEqual

```

9  [C#] public static SqlBoolean GreaterThanOrEqual(SqlMoney x, SqlMoney y);
10 [C++] public: static SqlBoolean GreaterThanOrEqual(SqlMoney x, SqlMoney y);
11 [VB] Public Shared Function GreaterThanOrEqual(ByVal x As SqlMoney, ByVal
12 y          As          SqlMoney)          As          SqlBoolean
13 [JScript] public static function GreaterThanOrEqual(x : SqlMoney, y : SqlMoney)
14 :                                          SqlBoolean;
    
```

*Description*

[ .]

LessThan

```

20 [C#] public static SqlBoolean LessThan(SqlMoney x, SqlMoney y);
21 [C++] public: static SqlBoolean LessThan(SqlMoney x, SqlMoney y);
22 [VB] Public Shared Function LessThan(ByVal x As SqlMoney, ByVal y As
23 SqlMoney)                                As                                SqlBoolean
24 [JScript] public static function LessThan(x : SqlMoney, y : SqlMoney) :
25 SqlBoolean;
    
```

*Description*

[ .]

LessThanOrEqualTo

[C#] public static SqlBoolean LessThanOrEqualTo(SqlMoney x, SqlMoney y);

[C++] public: static SqlBoolean LessThanOrEqualTo(SqlMoney x, SqlMoney y);

[VB] Public Shared Function LessThanOrEqualTo(ByVal x As SqlMoney, ByVal y

As SqlMoney) As SqlBoolean

[JScript] public static function LessThanOrEqualTo(x : SqlMoney, y : SqlMoney) :

SqlBoolean;

*Description*

[ .]

Multiply

[C#] public static SqlMoney Multiply(SqlMoney x, SqlMoney y);

[C++] public: static SqlMoney Multiply(SqlMoney x, SqlMoney y);

[VB] Public Shared Function Multiply(ByVal x As SqlMoney, ByVal y As

SqlMoney) As SqlMoney

[JScript] public static function Multiply(x : SqlMoney, y : SqlMoney) : SqlMoney;

*Description*

[ .]

NotEquals

```

1
2 [C#] public static SqlBoolean NotEquals(SqlMoney x, SqlMoney y);
3 [C++] public: static SqlBoolean NotEquals(SqlMoney x, SqlMoney y);
4 [VB] Public Shared Function NotEquals(ByVal x As SqlMoney, ByVal y As
5 SqlMoney) As SqlBoolean
6 [JScript] public static function NotEquals(x : SqlMoney, y : SqlMoney) :
7 SqlBoolean;
8

```

9 *Description*

```

10 [ .]
11 op_Addition
12

```

```

13 [C#] public static SqlMoney operator +(SqlMoney x, SqlMoney y);
14 [C++] public: static SqlMoney op_Addition(SqlMoney x, SqlMoney y);
15 [VB] returnValue = SqlMoney.op_Addition(x, y)
16 [JScript] returnValue = x + y;
17

```

18 *Description*

19 Calculates the sum of the two **System.Data.SqlTypes.SqlMoney**  
20 parameters.

21 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
22 **System.Data.SqlTypes.SqlMoney.Value** contains the sum of the two  
23 **System.Data.SqlTypes.SqlMoney** parameters. A  
24 **System.Data.SqlTypes.SqlMoney** structure. A  
25 **System.Data.SqlTypes.SqlMoney** structure.

op\_Division

[C#] public static SqlMoney operator /(SqlMoney x, SqlMoney y);

[C++] public: static SqlMoney op\_Division(SqlMoney x, SqlMoney y);

[VB] returnValue = SqlMoney.op\_Division(x, y)

[JScript] returnValue = x / y;

### Description

The division operator divides the first **System.Data.SqlTypes.SqlMoney** parameter by the second.

**Return Value:** A new **System.Data.SqlTypes.SqlMoney** structure whose **System.Data.SqlTypes.SqlMoney.Value** contains the results of the division. A **System.Data.SqlTypes.SqlMoney** structure. A **System.Data.SqlTypes.SqlMoney** structure.

op\_Equality

[C#] public static SqlBoolean operator ==(SqlMoney x, SqlMoney y);

[C++] public: static SqlBoolean op\_Equality(SqlMoney x, SqlMoney y);

[VB] returnValue = SqlMoney.op\_Equality(x, y)

[JScript] returnValue = x == y;

### Description

Performs a logical comparison of the two **System.Data.SqlTypes.SqlMoney** parameters to determine if they are equal.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is

1 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or  
 2 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If  
 3 either instance of **System.Data.SqlTypes.SqlMoney** is null, the  
 4 **System.Data.SqlTypes.SqlBoolean.Value** of the  
 5 **System.Data.SqlTypes.SqlBoolean** will be  
 6 **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlMoney**  
 7 structure. A **System.Data.SqlTypes.SqlMoney** structure.

8 **op\_Explicit**

9  
 10 [C#] public static explicit operator SqlMoney(SqlBoolean x);

11 [C++] public: static SqlMoney op\_Explicit(SqlBoolean x);

12 [VB] returnValue = SqlMoney.op\_Explicit(x)

13 [JScript] returnValue = SqlMoney(x);

14  
 15 *Description*

16 This implicit operator converts the supplied **System.Data.SqlTypes.SqlBit**  
 17 parameter to **System.Data.SqlTypes.SqlMoney**.

18 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
 19 **System.Data.SqlTypes.SqlMoney.Value** property equals the  
 20 **System.Data.SqlTypes.SqlBit.ByteValue** property of the  
 21 **System.Data.SqlTypes.SqlBit** parameter. The **System.Data.SqlTypes.SqlBit**  
 22 structure to be converted.

23 **op\_Explicit**

24  
 25 [C#] public static explicit operator SqlMoney(SqlDecimal x);

1	[C++]	public: static	SqlMoney	op_Explicit(SqlDecimal x);
2	[VB]	returnValue	=	SqlMoney.op_Explicit(x)
3	[JScript]	returnValue	=	SqlMoney(x);

5 *Description*

6 This operator converts the supplied **System.Data.SqlTypes.SqlDecimal**  
7 parameter to **System.Data.SqlTypes.SqlMoney**.

8 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
9 **System.Data.SqlTypes.SqlMoney.Value** property equals the  
10 **System.Data.SqlTypes.SqlDecimal.Value** of the  
11 **System.Data.SqlTypes.SqlDecimal** parameter. The  
12 **System.Data.SqlTypes.SqlDecimal** structure to be converted.

13 op Explicit

15	[C#]	public	static	explicit	operator	SqlMoney(SqlDouble	x);
16	[C++]	public:	static	SqlMoney	op_Explicit(SqlDouble	x);	
17	[VB]	returnValue	=	SqlMoney.op_Explicit(x)			
18	[JScript]	returnValue	=	SqlMoney(x);			

20	Description
----	-------------

21 This operator converts the supplied **System.Data.SqlTypes.SqlDouble**  
22 parameter to **System.Data.SqlTypes.SqlMoney**.

23 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
24 **System.Data.SqlTypes.SqlMoney.Value** property equals the  
25 **System.Data.SqlTypes.SqlDouble.Value** of the



1        This operator converts the supplied **System.Data.SqlTypes.SqlSingle**  
 2 parameter            to            **System.Data.SqlTypes.SqlMoney** .  
 3 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
 4 **System.Data.SqlTypes.SqlMoney.Value** property equals the  
 5 **System.Data.SqlTypes.SqlSingle.Value** of the **System.Data.SqlTypes.SqlSingle**  
 6 parameter. The **System.Data.SqlTypes.SqlSingle** structure to be converted.

7        op\_Explicit

8  
 9 [C#]    public    static    explicit    operator    SqlMoney(SqlString    x);  
 10 [C++]    public:    static    SqlMoney    op\_Explicit(SqlString    x);  
 11 [VB]            returnValue            =            SqlMoney.op\_Explicit(x)  
 12 [JScript]            returnValue            =            SqlMoney(x);

13  
 14 *Description*

15        This operator converts the **System.Data.SqlTypes.SqlString** parameter to  
 16 **System.Data.SqlTypes.SqlMoney** .  
 17 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
 18 **System.Data.SqlTypes.SqlMoney.Value** property equals the value represented  
 19 by the **System.Data.SqlTypes.SqlString** parameter. The  
 20 **System.Data.SqlTypes.SqlString** object to be converted.

21        op\_GreaterThan

22  
 23 [C#]    public    static    SqlBoolean    operator    >(SqlMoney x, SqlMoney y);  
 24 [C++]    public:    static    SqlBoolean    op\_GreaterThan(SqlMoney x, SqlMoney y);  
 25 [VB]            returnValue            =            SqlMoney.op\_GreaterThan(x,            y)



than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlMoney** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlMoney** structure. A **System.Data.SqlTypes.SqlMoney** structure.

op\_Explicit

[C#] public static implicit operator SqlMoney(decimal x);

[C++] public: static SqlMoney op\_Explicit(Decimal x);

[VB] returnValue = SqlMoney.op\_Explicit(x)

[JScript] returnValue = x;

### Description

Converts the **System.Decimal** parameter to **System.Data.SqlTypes.SqlMoney**.

*Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose **System.Data.SqlTypes.SqlMoney.Value** equals the value of the **System.Decimal** parameter.

op\_Explicit

[C#] public static implicit operator SqlMoney(SqlByte x);

```

1  [C++]      public:      static      SqlMoney      op_Implicit(SqlByte      x);
2  [VB]              returnValue      =      SqlMoney.op_Implicit(x)
3  [JScript]              returnValue      =      x;

```

4  
5 *Description*

6       This       implicit       operator       converts       the       supplied  
7 **System.Data.SqlTypes.SqlByte** parameter to **System.Data.SqlTypes.SqlMoney**

8 .  
9 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
10 **System.Data.SqlTypes.SqlMoney.Value** property is equal to the  
11 **System.Data.SqlTypes.SqlByte.Value** of the **System.Data.SqlTypes.SqlByte**  
12 parameter. The **System.Data.SqlTypes.SqlByte** structure to be converted.

13       op\_Implicit

```

14  
15 [C#]      public      static      implicit      operator      SqlMoney(SqlInt16      x);
16 [C++]      public:      static      SqlMoney      op_Implicit(SqlInt16      x);
17 [VB]              returnValue      =      SqlMoney.op_Implicit(x)
18 [JScript]              returnValue      =      x;

```

19  
20 *Description*

21       This       implicit       operator       converts       the       supplied  
22 **System.Data.SqlTypes.SqlInt16** parameter to **System.Data.SqlTypes.SqlMoney**

23 .  
24 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
25 **System.Data.SqlTypes.SqlMoney.Value** property equals the

1 **System.Data.SqlTypes.SqlInt16.Value** of the **System.Data.SqlTypes.SqlInt16**  
 2 parameter. The **System.Data.SqlTypes.SqlInt16** structure to be converted.

3 **op\_Implicit**

4  
 5 [C#] public static implicit operator SqlMoney(SqlInt32 x);  
 6 [C++] public: static SqlMoney op\_Implicit(SqlInt32 x);  
 7 [VB] returnValue = SqlMoney.op\_Implicit(x)  
 8 [JScript] returnValue = x;

9  
 10 *Description*

11 This implicit operator converts the supplied  
 12 **System.Data.SqlTypes.SqlInt32** parameter to **System.Data.SqlTypes.SqlMoney**

13 .  
 14 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose  
 15 **System.Data.SqlTypes.SqlMoney.Value** property equals the  
 16 **System.Data.SqlTypes.SqlInt32.Value** of the **System.Data.SqlTypes.SqlInt32**  
 17 parameter. The **System.Data.SqlTypes.SqlInt32** structure to be converted.

18 **op\_Implicit**

19  
 20 [C#] public static implicit operator SqlMoney(SqlInt64 x);  
 21 [C++] public: static SqlMoney op\_Implicit(SqlInt64 x);  
 22 [VB] returnValue = SqlMoney.op\_Implicit(x)  
 23 [JScript] returnValue = x;

24  
 25 *Description*

This implicit operator converts the supplied **System.Data.SqlTypes.SqlInt64** parameter to **System.Data.SqlTypes.SqlMoney**

*Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure whose **System.Data.SqlTypes.SqlMoney.Value** property equals the **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64** parameter. The **System.Data.SqlTypes.SqlInt64** structure to be converted.

op\_Inequality

[C#] public static SqlBoolean operator !=(SqlMoney x, SqlMoney y);

[C++] public: static SqlBoolean op\_Inequality(SqlMoney x, SqlMoney y);

[VB] returnValue = SqlMoney.op\_Inequality(x, y)

[JScript] returnValue = x != y;

### Description

Performs a logical comparison of the two **System.Data.SqlTypes.SqlMoney** parameters to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **System.Data.SqlTypes.SqlMoney** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlMoney** structure. A **System.Data.SqlTypes.SqlMoney** structure.

```

1      op_LessThan
2
3  [C#]      public      static      SqlBoolean      operator
4  [C++] public: static SqlBoolean op_LessThan(SqlMoney x, SqlMoney y);
5  [VB]      returnValue      =      SqlMoney.op_LessThan(x,      y)
6  [JScript]      returnValue      =      x      <      y;

```

### Description

Performs a logical comparison of the two **System.Data.SqlTypes.SqlMoney** parameters to determine if the first is less than the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **System.Data.SqlTypes.SqlMoney** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlMoney** structure. A **System.Data.SqlTypes.SqlMoney** structure.

```

20     op_LessThanOrEqual
21
22  [C#] public static SqlBoolean operator <=(SqlMoney x, SqlMoney y);
23  [C++] public: static SqlBoolean op_LessThanOrEqual(SqlMoney x, SqlMoney y);
24  [VB]      returnValue      =      SqlMoney.op_LessThanOrEqual(x,      y)
25  [JScript]      returnValue      =      x      <=      y;

```

## Description

Performs a logical comparison of the two **System.Data.SqlTypes.SqlMoney** parameters to determine if the first is less than or equal to the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlMoney** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlMoney** structure. A **System.Data.SqlTypes.SqlMoney** structure.

op\_Multiply

[C#] public static SqlMoney operator \*(SqlMoney x, SqlMoney y);

[C++] public: static SqlMoney op\_Multiply(SqlMoney x, SqlMoney y);

[VB] returnValue = SqlMoney.op\_Multiply(x, y)

[JScript] returnValue = x \* y;

## Description

The multiplication operator calculates the product of the two **System.Data.SqlTypes.SqlMoney** parameters.

**Return Value:** A new **System.Data.SqlTypes.SqlMoney** structure whose **System.Data.SqlTypes.SqlMoney.Value** contains the product of the

1 multiplication. A **System.Data.SqlTypes.SqlMoney** structure. A  
2 **System.Data.SqlTypes.SqlMoney** structure.

3 op\_Subtraction

5 [C#] public static **SqlMoney** operator -(**SqlMoney** x, **SqlMoney** y);

6 [C++] public: static **SqlMoney** op\_Subtraction(**SqlMoney** x, **SqlMoney** y);

7 [VB] returnValue = **SqlMoney.op\_Subtraction**(x, y)

8 [JScript] returnValue = x - y;

10 *Description*

11 The subtraction operator subtracts the second  
12 **System.Data.SqlTypes.SqlMoney** parameter from the first.

13 *Return Value:* A new **System.Data.SqlTypes.SqlMoney** structure containing the  
14 results of the subtraction. A **System.Data.SqlTypes.SqlMoney** structure. A  
15 **System.Data.SqlTypes.SqlMoney** structure.

16 op\_UnaryNegation

18 [C#] public static **SqlMoney** operator -(**SqlMoney** x);

19 [C++] public: static **SqlMoney** op\_UnaryNegation(**SqlMoney** x);

20 [VB] returnValue = **SqlMoney.op\_UnaryNegation**(x)

21 [JScript] returnValue = -x;

23 *Description*

24 The unary minus operator negates the **System.Data.SqlTypes.SqlMoney**  
25 parameter.

1 *Return Value:* A **System.Data.SqlTypes.SqlMoney** structure whose  
 2 **System.Data.SqlTypes.SqlMoney.Value** contains the results of the negation. The  
 3 **System.Data.SqlTypes.SqlMoney** structure to be negated.

4 Parse

5  
 6 [C#] public static SqlMoney Parse(string s);  
 7 [C++] public: static SqlMoney Parse(String\* s);  
 8 [VB] Public Shared Function Parse(ByVal s As String) As SqlMoney  
 9 [JScript] public static function Parse(s : String) : SqlMoney;

10  
 11 *Description*

12 [ .][ .]

13 Subtract

14  
 15 [C#] public static SqlMoney Subtract(SqlMoney x, SqlMoney y);  
 16 [C++] public: static SqlMoney Subtract(SqlMoney x, SqlMoney y);  
 17 [VB] Public Shared Function Subtract(ByVal x As SqlMoney, ByVal y As  
 18 SqlMoney) As SqlMoney  
 19 [JScript] public static function Subtract(x : SqlMoney, y : SqlMoney) : SqlMoney;

20  
 21 *Description*

22 [ .]

23 ToDecimal

24  
 25 [C#] public decimal ToDecimal();

```

1  [C++]          public:          Decimal          ToDecimal();
2  [VB]          Public          Function          ToDecimal()          As          Decimal
3  [JScript]          public          function          ToDecimal()          :          Decimal;

```

#### 5 *Description*

6 Converts the Value of this instance of **System.Data.SqlTypes.SqlMoney**  
7 as a **System.Decimal** structure.

8 *Return Value:* A **System.Decimal** structure whose value equals the  
9 **System.Data.SqlTypes.SqlMoney.Value** property of this  
10 **System.Data.SqlTypes.SqlMoney** structure.

#### 11 **ToDouble**

```

13 [C#]          public          double          ToDouble();
14 [C++]          public:          double          ToDouble();
15 [VB]          Public          Function          ToDouble()          As          Double
16 [JScript]          public          function          ToDouble()          :          double;

```

#### 18 *Description*

19 Converts this **System.Data.SqlTypes.SqlMoney** structure to a double.  
20 *Return Value:* A double with a value equal to this  
21 **System.Data.SqlTypes.SqlMoney** structure.

#### 22 **ToInt32**

```

24 [C#]          public          int          ToInt32();
25 [C++]          public:          int          ToInt32();

```

1	[VB]	Public	Function	ToInt32()	As	Integer
2	[JScript]	public	function	ToInt32()	:	int;

3

4 *Description*

5 Converts this **System.Data.SqlTypes.SqlMoney** structure to integer.

6 *Return Value:* A 32-bit integer whose value equals the integer portion of this  
 7 **System.Data.SqlTypes.SqlMoney** structure.

8 ToInt64

9

10	[C#]	public	long	ToInt64();
----	------	--------	------	------------

11	[C++]	public:	__int64	ToInt64();
----	-------	---------	---------	------------

12	[VB]	Public	Function	ToInt64()	As	Long
----	------	--------	----------	-----------	----	------

13	[JScript]	public	function	ToInt64()	:	long;
----	-----------	--------	----------	-----------	---	-------

14

15 *Description*

16 Converts the Value of this **System.Data.SqlTypes.SqlMoney** structure to  
 17 long.

18 *Return Value:* A 64-bit integer whose value equals the integer portion of this  
 19 **System.Data.SqlTypes.SqlMoney** structure.

20 ToSqlBoolean

21

22	[C#]	public	SqlBoolean	ToSqlBoolean();
----	------	--------	------------	-----------------

23	[C++]	public:	SqlBoolean	ToSqlBoolean();
----	-------	---------	------------	-----------------

24	[VB]	Public	Function	ToSqlBoolean()	As	SqlBoolean
----	------	--------	----------	----------------	----	------------

25	[JScript]	public	function	ToSqlBoolean()	:	SqlBoolean;
----	-----------	--------	----------	----------------	---	-------------

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

[ .]

ToSqlByte

[C#]	public		SqlByte		ToSqlByte();
[C++]	public:		SqlByte		ToSqlByte();
[VB]	Public	Function	ToSqlByte()	As	SqlByte
[JScript]	public	function	ToSqlByte()	:	SqlByte;

*Description*

[ .]

ToSqlDecimal

[C#]	public		SqlDecimal		ToSqlDecimal();
[C++]	public:		SqlDecimal		ToSqlDecimal();
[VB]	Public	Function	ToSqlDecimal()	As	SqlDecimal
[JScript]	public	function	ToSqlDecimal()	:	SqlDecimal;

*Description*

[ .]

ToSqlDouble

[C#]	public		SqlDouble		ToSqlDouble();
[C++]	public:		SqlDouble		ToSqlDouble();

1 [VB] Public Function ToSqlDouble() As SqlDouble  
 2 [JScript] public function ToSqlDouble() : SqlDouble;

3  
 4 *Description*

5 [ .]  
 6 ToSqlInt16

8 [C#] public SqlInt16 ToSqlInt16();

9 [C++] public: SqlInt16 ToSqlInt16();

10 [VB] Public Function ToSqlInt16() As SqlInt16

11 [JScript] public function ToSqlInt16() : SqlInt16;

12  
 13 *Description*

14 [ .]  
 15 ToSqlInt32

17 [C#] public SqlInt32 ToSqlInt32();

18 [C++] public: SqlInt32 ToSqlInt32();

19 [VB] Public Function ToSqlInt32() As SqlInt32

20 [JScript] public function ToSqlInt32() : SqlInt32;

21  
 22 *Description*

23 [ .]  
 24 ToSqlInt64

25

1					
2	[C#]	public	SqlInt64	ToSqlInt64();	
3	[C++]	public:	SqlInt64	ToSqlInt64();	
4	[VB]	Public	Function	ToSqlInt64()	As SqlInt64
5	[JScript]	public	function	ToSqlInt64()	: SqlInt64;

6

7 *Description*

8 [ .]

9 ToSqlSingle

11	[C#]	public	SqlSingle	ToSqlSingle();	
12	[C++]	public:	SqlSingle	ToSqlSingle();	
13	[VB]	Public	Function	ToSqlSingle()	As SqlSingle
14	[JScript]	public	function	ToSqlSingle()	: SqlSingle;

15

16 *Description*

17 [ .]

18 ToSqlString

20	[C#]	public	SqlString	ToSqlString();	
21	[C++]	public:	SqlString	ToSqlString();	
22	[VB]	Public	Function	ToSqlString()	As SqlString
23	[JScript]	public	function	ToSqlString()	: SqlString;

24

25 *Description*

1 [ .]  
 2 ToString  
 3  
 4 [C#] public override string ToString();  
 5 [C++] public: String\* ToString();  
 6 [VB] Overrides Public Function ToString() As String  
 7 [JScript] public override function ToString() : String; Converts a  
 8 **System.Data.SqlTypes.SqlMoney** structure to string.

#### 10 *Description*

11 Converts this instance of **System.Data.SqlTypes.SqlMoney** to string.  
 12 *Return Value:* A string whose value is the string representation of the  
 13 **System.Data.SqlTypes.SqlMoney.Value** property of this  
 14 **System.Data.SqlTypes.SqlMoney** structure.

15 SqlNullValueException class (System.Data.SqlTypes)

16 ToString

#### 19 *Description*

20 The exception that is thrown when the **Value** property of a **SqlTypes**  
 21 structure is set to null.

22 In order to avoid throwing this exception, you should always check the  
 23 **IsNull** property of the structure before accessing the **Value** property.

24 SqlNullValueException

25 *Example Syntax:*

1 ToString

2

3 [C#] public SqlNullValueException();

4 [C++] public: SqlNullValueException();

5 [VB] Public Sub New()

6 [JScript] public function SqlNullValueException(); Initializes a new instance of

7 the **System.Data.SqlTypes.SqlNullValueException** class.

8

9 *Description*

10 Initializes a new instance of the

11 **System.Data.SqlTypes.SqlNullValueException** class with default properties.

12 SqlNullValueException

13 *Example Syntax:*

14 ToString

15

16 [C#] public SqlNullValueException(string message);

17 [C++] public: SqlNullValueException(String\* message);

18 [VB] Public Sub New(ByVal message As String)

19 [JScript] public function SqlNullValueException(message : String);

20

21 *Description*

22 Initializes a new instance of the

23 **System.Data.SqlTypes.SqlNullValueException** class with a specified error

24 message. The error message that explains the reason for the exception.

25 HelpLink

```

1      HRESULT
2      InnerException
3      Message
4      Source
5      StackTrace
6      TargetSite
7      ISerializable.GetObjectData
8
9      [C#] void ISerializable.GetObjectData(SerializationInfo si, StreamingContext
10     context);
11     [C++] void ISerializable::GetObjectData(SerializationInfo* si, StreamingContext
12     context);
13     [VB] Sub GetObjectData(ByVal si As SerializationInfo, ByVal context As
14     StreamingContext) Implements ISerializable.GetObjectData
15     [JScript] function ISerializable.GetObjectData(si : SerializationInfo, context :
16     StreamingContext);
17
18     SqlSingle structure (System.Data.SqlTypes)
19
20     ToString
21
22     Description
23
24     Represents a floating point number within the range of -3.40E +38 through
25     3.40E +38 to be stored in or retrieved from a database.
26
27     ToString

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      static      readonly      SqlSingle      MaxValue;
[C++]      public:      static      SqlSingle      MaxValue;
[VB]      Public      Shared      ReadOnly      MaxValue      As      SqlSingle
[JScript]      public      static      var      MaxValue      :      SqlSingle;
```

*Description*

Represents the maximum value that can be assigned to the **System.Data.SqlTypes.SqlSingle.Value** property of an instance of the **System.Data.SqlTypes.SqlSingle** class.

The value of this constant is -3.40E+38.

ToString

```
[C#]      public      static      readonly      SqlSingle      MinValue;
[C++]      public:      static      SqlSingle      MinValue;
[VB]      Public      Shared      ReadOnly      MinValue      As      SqlSingle
[JScript]      public      static      var      MinValue      :      SqlSingle;
```

*Description*

Represents the minimum value that can be assigned to the **System.Data.SqlTypes.SqlSingle.Value** property of an instance of the **System.Data.SqlTypes.SqlSingle** class.

The value of this constant is 3.40E+38.

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]      public      static      readonly      SqlSingle      Null;
[C++]      public:      static      SqlSingle      Null;
[VB]      Public      Shared      ReadOnly      Null      As      SqlSingle
[JScript]      public      static      var      Null      :      SqlSingle;
```

*Description*

[ .]  
ToString

```
[C#]      public      static      readonly      SqlSingle      Zero;
[C++]      public:      static      SqlSingle      Zero;
[VB]      Public      Shared      ReadOnly      Zero      As      SqlSingle
[JScript]      public      static      var      Zero      :      SqlSingle;
```

*Description*

Represents the zero value that can be assigned to the **System.Data.SqlTypes.SqlSingle.Value** property of an instance of the **System.Data.SqlTypes.SqlSingle** class.

**System.Data.SqlTypes.SqlSingle.Zero** functions as a constant for the **System.Data.SqlTypes.SqlSingle** class.

SqlSingle

*Example Syntax:*

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]          public          SqlSingle(double          value);
[C++]          public:          SqlSingle(double          value);
[VB]      Public      Sub      New(ByVal      value      As      Double)
[JScript]      public      function      SqlSingle(value      :      double);
```

*Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlSingle** structure using the supplied double parameter. A double value which will be used as the **System.Data.SqlTypes.SqlSingle.Value** of the new **System.Data.SqlTypes.SqlSingle** structure.

SqlSingle

*Example Syntax:*

ToString

```
[C#]          public          SqlSingle(float          value);
[C++]          public:          SqlSingle(float          value);
[VB]      Public      Sub      New(ByVal      value      As      Single)
[JScript] public function SqlSingle(value : float); Initializes a new instance of the
System.Data.SqlTypes.SqlSingle structure using the supplied floating point
value.
```

*Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlSingle** structure. A floating point number which will be used as the

1 **System.Data.SqlTypes.SqlSingle.Value** of the new  
2 **System.Data.SqlTypes.SqlSingle** structure.

3 IsNull

4 ToString

6 [C#] public bool IsNull {get;}

7 [C++] public: \_\_property bool get\_IsNull();

8 [VB] Public ReadOnly Property IsNull As Boolean

9 [JScript] public function get IsNull() : Boolean;

11 *Description*

12 Returns a value indicating whether the  
13 **System.Data.SqlTypes.SqlSingle.Value** property is assigned to null.

14 Value

15 ToString

17 [C#] public float Value {get;}

18 [C++] public: \_\_property float get\_Value();

19 [VB] Public ReadOnly Property Value As Single

20 [JScript] public function get Value() : float;

22 *Description*

23 Gets the value of this **System.Data.SqlTypes.SqlSingle** structure. This  
24 property is read-only.

25 Add

```

1
2 [C#] public static SqlSingle Add(SqlSingle x, SqlSingle y);
3 [C++] public: static SqlSingle Add(SqlSingle x, SqlSingle y);
4 [VB] Public Shared Function Add(ByVal x As SqlSingle, ByVal y As SqlSingle)
5 As SqlSingle
6 [JScript] public static function Add(x : SqlSingle, y : SqlSingle) : SqlSingle;
7

```

### Description

[ .]

CompareTo

```

11
12 [C#] public int CompareTo(object value);
13 [C++] public: __sealed int CompareTo(Object* value);
14 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
15 Integer
16 [JScript] public function CompareTo(value : Object) : int;
17

```

### Description

Compares this instance to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

Divide

```

23
24
25 [C#] public static SqlSingle Divide(SqlSingle x, SqlSingle y);

```

```

1  [C++] public: static SqlSingle Divide(SqlSingle x, SqlSingle y);
2  [VB] Public Shared Function Divide(ByVal x As SqlSingle, ByVal y As
3  SqlSingle) As SqlSingle
4  [JScript] public static function Divide(x : SqlSingle, y : SqlSingle) : SqlSingle;

```

5  
6 *Description*

7 [ .]  
8 Equals

```

9
10 [C#] public override bool Equals(object value);
11 [C++] public: bool Equals(Object* value);
12 [VB] Overrides Public Function Equals(ByVal value As Object) As Boolean
13 [JScript] public override function Equals(value : Object) : Boolean;

```

14  
15 *Description*

16 Compares the supplied object parameter to the  
17 **System.Data.SqlTypes.SqlSingle.Value** property of the  
18 **System.Data.SqlTypes.SqlSingle** object.

19 *Return Value:* Equals will return **true** if the object is an instance of  
20 **System.Data.SqlTypes.SqlSingle** and the two are equal; otherwise **false** . The  
21 object to be compared.

22 Equals

```

23
24 [C#] public static new SqlBoolean Equals(SqlSingle x, SqlSingle y);
25 [C++] public: static SqlBoolean Equals(SqlSingle x, SqlSingle y);

```

1 [VB] Shadows Public Shared Function Equals(ByVal x As SqlSingle, ByVal y As  
2 SqlSingle) As SqlBoolean  
3 [JScript] public static hide function Equals(x : SqlSingle, y : SqlSingle) :  
4 SqlBoolean;

5  
6 *Description*

7 [ .]  
8 GetHashCode

9  
10 [C#] public override int GetHashCode();  
11 [C++] public: int GetHashCode();  
12 [VB] Overrides Public Function GetHashCode() As Integer  
13 [JScript] public override function GetHashCode() : int;

14  
15 *Description*

16 Gets the hash code for this instance.

17 *Return Value:* A 32-bit signed integer hash code.

18 GreaterThan

19  
20 [C#] public static SqlBoolean GreaterThan(SqlSingle x, SqlSingle y);  
21 [C++] public: static SqlBoolean GreaterThan(SqlSingle x, SqlSingle y);  
22 [VB] Public Shared Function GreaterThan(ByVal x As SqlSingle, ByVal y As  
23 SqlSingle) As SqlBoolean  
24 [JScript] public static function GreaterThan(x : SqlSingle, y : SqlSingle) :  
25 SqlBoolean;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

[ .]

GreaterThanOrEqualTo

```
[C#] public static SqlBoolean GreaterThanOrEqualTo(SqlSingle x, SqlSingle y);
[C++] public: static SqlBoolean GreaterThanOrEqualTo(SqlSingle x, SqlSingle y);
[VB] Public Shared Function GreaterThanOrEqualTo(ByVal x As SqlSingle, ByVal
y          As          SqlSingle)          As          SqlBoolean
[JScript] public static function GreaterThanOrEqualTo(x : SqlSingle, y : SqlSingle) :
SqlBoolean;
```

*Description*

[ .]

LessThan

```
[C#] public static SqlBoolean LessThan(SqlSingle x, SqlSingle y);
[C++] public: static SqlBoolean LessThan(SqlSingle x, SqlSingle y);
[VB] Public Shared Function LessThan(ByVal x As SqlSingle, ByVal y As
SqlSingle)          As          SqlBoolean
[JScript] public static function LessThan(x : SqlSingle, y : SqlSingle) :
SqlBoolean;
```

*Description*

[ .]

## LessThanOrEqual

```
[C#] public static SqlBoolean LessThanOrEqual(SqlSingle x, SqlSingle y);  
[C++] public: static SqlBoolean LessThanOrEqual(SqlSingle x, SqlSingle y);  
[VB] Public Shared Function LessThanOrEqual(ByVal x As SqlSingle, ByVal y  
As SqlSingle) As SqlBoolean  
[JScript] public static function LessThanOrEqual(x : SqlSingle, y : SqlSingle) :  
SqlBoolean;
```

### *Description*

[ .]

Multiply

```
[C#] public static SqlSingle Multiply(SqlSingle x, SqlSingle y);  
[C++] public: static SqlSingle Multiply(SqlSingle x, SqlSingle y);  
[VB] Public Shared Function Multiply(ByVal x As SqlSingle, ByVal y As  
SqlSingle) As SqlSingle  
[JScript] public static function Multiply(x : SqlSingle, y : SqlSingle) : SqlSingle;
```

### *Description*

[ .]

NotEquals

```
[C#] public static SqlBoolean NotEquals(SqlSingle x, SqlSingle y);  
[C++] public: static SqlBoolean NotEquals(SqlSingle x, SqlSingle y);
```

```

1 [VB] Public Shared Function NotEquals(ByVal x As SqlSingle, ByVal y As
2 SqlSingle) As SqlBoolean
3 [JScript] public static function NotEquals(x : SqlSingle, y : SqlSingle) :
4 SqlBoolean;

```

5  
6 *Description*

```

7 [ .]
8 op_Addition

```

```

9
10 [C#] public static SqlSingle operator +(SqlSingle x, SqlSingle y);
11 [C++] public: static SqlSingle op_Addition(SqlSingle x, SqlSingle y);
12 [VB] returnValue = SqlSingle.op_Addition(x, y)
13 [JScript] returnValue = x + y;

```

14  
15 *Description*

```

16 [ .] [ .] A System.Data.SqlTypes.SqlSingle structure. A
17 System.Data.SqlTypes.SqlSingle structure.

```

```

18 op_Division

```

```

19
20 [C#] public static SqlSingle operator /(SqlSingle x, SqlSingle y);
21 [C++] public: static SqlSingle op_Division(SqlSingle x, SqlSingle y);
22 [VB] returnValue = SqlSingle.op_Division(x, y)
23 [JScript] returnValue = x / y;

```

24  
25 *Description*

[ .] [ .] A **System.Data.SqlTypes.SqlSingle** structure. A **System.Data.SqlTypes.SqlSingle** structure.

op\_Equality

[C#] public static SqlBoolean operator ==(SqlSingle x, SqlSingle y);

[C++] public: static SqlBoolean op\_Equality(SqlSingle x, SqlSingle y);

[VB] returnValue = SqlSingle.op\_Equality(x, y)

[JScript] returnValue = x == y;

### *Description*

Performs a logical comparison of the two **SqlSingle** parameters to determine if they are equal.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If either instance of **System.Data.SqlTypes.SqlSingle** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlSingle** structure. A **System.Data.SqlTypes.SqlSingle** structure.

op\_Explicit

[C#] public static explicit operator SqlSingle(SqlBoolean x);

[C++] public: static SqlSingle op\_Explicit(SqlBoolean x);

[VB] returnValue = SqlSingle.op\_Explicit(x)

[JScript]                      returnValue                      =                      SqlSingle(x);

*Description*

This implicit operator converts the supplied **System.Data.SqlTypes.SqlBit** to **System.Data.SqlTypes.SqlSingle**.

*Return Value:* A new **System.Data.SqlTypes.SqlSingle** structure whose **System.Data.SqlTypes.SqlSingle.Value** is equal to the **System.Data.SqlTypes.SqlBit.ByteValue** of the **System.Data.SqlTypes.SqlBit** parameter. The **System.Data.SqlTypes.SqlBit** structure to be converted.

op\_Explicit

[C#]      public      static      explicit      operator      SqlSingle(SqlDouble      x);

[C++]      public:      static      SqlSingle      op\_Explicit(SqlDouble      x);

[VB]                      returnValue                      =                      SqlSingle.op\_Explicit(x)

[JScript]                      returnValue                      =                      SqlSingle(x);

*Description*

Converts the supplied **System.Data.SqlTypes.SqlDouble** parameter to **System.Data.SqlTypes.SqlSingle**.

*Return Value:* A new **System.Data.SqlTypes.SqlSingle** structure whose **System.Data.SqlTypes.SqlSingle.Value** is equal to the **System.Data.SqlTypes.SqlDouble.Value** of the **System.Data.SqlTypes.SqlDouble** parameter. The **System.Data.SqlTypes.SqlDouble** parameter to be converted.

op\_Explicit



```

1  [C++] public: static SqlBoolean op_GreaterThan(SqlSingle x, SqlSingle y);
2  [VB]      returnValue      =      SqlSingle.op_GreaterThan(x,      y)
3  [JScript]      returnValue      =      x      >      y;

```

4  
5 *Description*

6        Performs a logical comparison of the two  
7 **System.Data.SqlTypes.SqlSingle** operands to determine if the first is greater than  
8 the second.

9 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
10 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the  
11 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
12 instance of **System.Data.SqlTypes.SqlSingle** is null, the  
13 **System.Data.SqlTypes.SqlBoolean.Value** of the  
14 **System.Data.SqlTypes.SqlBoolean** will be  
15 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlSingle**  
16 structure. A **System.Data.SqlTypes.SqlSingle** structure.

17        op\_GreaterThanOrEqual

18  
19 [C#] public static SqlBoolean operator >=(SqlSingle x, SqlSingle y);  
20 [C++] public: static SqlBoolean op\_GreaterThanOrEqual(SqlSingle x, SqlSingle  
21 y);

```

22 [VB]      returnValue      =      SqlSingle.op_GreaterThanOrEqual(x,      y)
23 [JScript]      returnValue      =      x      >=      y;

```

24  
25 *Description*

Performs a logical comparison of two **System.Data.SqlTypes.SqlSingle** structures to determine if the first is greater than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlSingle** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlSingle** structure. A **System.Data.SqlTypes.SqlSingle** structure.

op\_Implicit

```
[C#] public static implicit operator SqlSingle(float x);
```

```
[C++] public: static SqlSingle op_Implicit(float x);
```

```
[VB]         returnValue = SqlSingle.op_Implicit(x)
```

```
[JScript]          returnValue          =          x;
```

### Description

$$[\cdot][\cdot]$$

**op\_Implicit**

```
[C#] public static implicit operator SqlSingle(SqlByte x);
```

```
[C++] public: static SqlSingle op_Implicit(SqlByte x);
```

```
[VB]         returnValue = SqlSingle.op_Implicit(x)
```

```
[JScript]      returnValue      =      x;
```

## Description

This implicit operator converts the **System.Data.SqlTypes.SqlByte** parameter to **System.Data.SqlTypes.SqlSingle**.

**Return Value:** A new **System.Data.SqlTypes.SqlSingle** structure whose **System.Data.SqlTypes.SqlSingle.Value** property equals the **System.Data.SqlTypes.SqlByte.Value** of the **System.Data.SqlTypes.SqlByte** parameter. The **System.Data.SqlTypes.SqlByte** to be converted.

op\_Implicit

[C#] public static implicit operator SqlSingle(SqlDecimal x);

[C++] public: static SqlSingle op\_Implicit(SqlDecimal x);

[VB] returnValue = SqlSingle.op\_Implicit(x)

[JScript] returnValue = x;

## Description

Converts the supplied **System.Data.SqlTypes.SqlDecimal** parameter to **System.Data.SqlTypes.SqlSingle**.

**Return Value:** A new **System.Data.SqlTypes.SqlSingle** structure whose **System.Data.SqlTypes.SqlSingle.Value** is equal to the **System.Data.SqlTypes.SqlDecimal.Value** of the **System.Data.SqlTypes.SqlDecimal** parameter. The **System.Data.SqlTypes.SqlDecimal** structure to be converted.

op\_Implicit

```

1
2 [C#]    public    static    implicit    operator    SqlSingle(SqlInt16    x);
3 [C++]    public:    static    SqlSingle    op_Implicit(SqlInt16    x);
4 [VB]        returnValue    =    SqlSingle.op_Implicit(x)
5 [JScript]        returnValue    =    x;
6

```

*Description*

Converts the supplied **System.Data.SqlTypes.SqlInt16** parameter to **System.Data.SqlTypes.SqlSingle**.

*Return Value:* A new **System.Data.SqlTypes.SqlSingle** structure whose **System.Data.SqlTypes.SqlSingle.Value** is equal to the **System.Data.SqlTypes.SqlInt16.Value** of the **System.Data.SqlTypes.SqlInt16** parameter. The **System.Data.SqlTypes.SqlInt16** structure to be converted.

```

14    op_Implicit
15
16 [C#]    public    static    implicit    operator    SqlSingle(SqlInt32    x);
17 [C++]    public:    static    SqlSingle    op_Implicit(SqlInt32    x);
18 [VB]        returnValue    =    SqlSingle.op_Implicit(x)
19 [JScript]        returnValue    =    x;
20

```

*Description*

Converts the supplied **System.Data.SqlTypes.SqlInt32** structure to **System.Data.SqlTypes.SqlSingle**.

*Return Value:* A new **System.Data.SqlTypes.SqlSingle** structure whose **System.Data.SqlTypes.SqlSingle.Value** is equal to the

1 **System.Data.SqlTypes.SqlInt32.Value** of the **System.Data.SqlTypes.SqlInt32**  
2 parameter. The **System.Data.SqlTypes.SqlInt32** structure to be converted.

3 **op\_Explicit**

4  
5 [C#] public static implicit operator SqlSingle(SqlInt64 x);

6 [C++] public: static SqlSingle op\_Explicit(SqlInt64 x);

7 [VB] returnValue = SqlSingle.op\_Explicit(x)

8 [JScript] returnValue = x;

9  
10 *Description*

11 Converts the supplied **System.Data.SqlTypes.SqlInt64** parameter to

12 **System.Data.SqlTypes.SqlSingle**

13 *Return Value:* A new **System.Data.SqlTypes.SqlSingle** structure whose

14 **System.Data.SqlTypes.SqlSingle.Value** is equal to the

15 **System.Data.SqlTypes.SqlInt64.Value** of the **System.Data.SqlTypes.SqlInt64**

16 parameter. The **System.Data.SqlTypes.SqlInt64** structure to be converted.

17 **op\_Explicit**

18  
19 [C#] public static implicit operator SqlSingle(SqlMoney x);

20 [C++] public: static SqlSingle op\_Explicit(SqlMoney x);

21 [VB] returnValue = SqlSingle.op\_Explicit(x)

22 [JScript] returnValue = x;

23  
24 *Description*

25

Converts the supplied **System.Data.SqlTypes.SqlMoney** structure to **System.Data.SqlTypes.SqlSingle**.

*Return Value:* A new **System.Data.SqlTypes.SqlSingle** structure whose **System.Data.SqlTypes.SqlSingle.Value** is equal to the **System.Data.SqlTypes.SqlMoney.Value** of the **System.Data.SqlTypes.SqlMoney** parameter. The **System.Data.SqlTypes.SqlMoney** structure to be converted.

**op\_Inequality**

[C#] public static SqlBoolean operator !=(SqlSingle x, SqlSingle y);

[C++] public: static SqlBoolean op\_Inequality(SqlSingle x, SqlSingle y);

[VB] returnValue = SqlSingle.op\_Inequality(x, y)

[JScript] returnValue = x != y;

#### *Description*

Performs a logical comparison of the two **System.Data.SqlTypes.SqlSingle** parameters to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **System.Data.SqlTypes.SqlSingle** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlSingle** structure. A **System.Data.SqlTypes.SqlSingle** structure.

op\_LessThan

```
[C#]          public          static          SqlBoolean          operator
[C++] public: static SqlBoolean op_LessThan(SqlSingle x, SqlSingle y);
[VB]          returnValue      =          SqlSingle.op_LessThan(x,          y)
[JScript]          returnValue      =          x          <          y;
```

#### *Description*

Performs a logical comparison of the two **System.Data.SqlTypes.SqlSingle** parameters to determine if the first is less than the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlSingle** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlSingle** structure. A **System.Data.SqlTypes.SqlSingle** structure.

op\_LessThanOrEqual

```
[C#] public static SqlBoolean operator <=(SqlSingle x, SqlSingle y);
[C++] public: static SqlBoolean op_LessThanOrEqual(SqlSingle x, SqlSingle y);
[VB]          returnValue      =          SqlSingle.op_LessThanOrEqual(x,          y)
[JScript]          returnValue      =          x          <=          y;
```

*Description*

Performs a logical comparison of the two **System.Data.SqlTypes.SqlSingle** parameters to determine if the first is less than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlSingle** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlSingle** structure. A **System.Data.SqlTypes.SqlSingle** structure.

op\_Multiply

[C#] public static SqlSingle operator \*(SqlSingle x, SqlSingle y);

[C++] public: static SqlSingle op\_Multiply(SqlSingle x, SqlSingle y);

[VB] returnValue = SqlSingle.op\_Multiply(x, y)

[JScript] returnValue = x \* y;

*Description*

[ ] [ ] A **System.Data.SqlTypes.SqlSingle** structure. A **System.Data.SqlTypes.SqlSingle** structure.

op\_Subtraction

```

1
2 [C#] public static SqlSingle operator -(SqlSingle x, SqlSingle y);
3 [C++] public: static SqlSingle op_Subtraction(SqlSingle x, SqlSingle y);
4 [VB]     returnValue          =          SqlSingle.op_Subtraction(x,          y)
5 [JScript]     returnValue          =          x          -          y;
6

```

*Description*

```

7
8     [ .] [ .] A System.Data.SqlTypes.SqlSingle structure. A
9 System.Data.SqlTypes.SqlSingle structure.

```

```

10     op_UnaryNegation

```

```

11
12 [C#]     public     static     SqlSingle     operator     -(SqlSingle     x);
13 [C++]     public:     static     SqlSingle     op_UnaryNegation(SqlSingle     x);
14 [VB]     returnValue          =          SqlSingle.op_UnaryNegation(x)
15 [JScript]     returnValue          =          -x;
16

```

*Description*

```

17
18     [ .] [ .] A System.Data.SqlTypes.SqlSingle structure.

```

```

19     Parse

```

```

20
21 [C#]     public     static     SqlSingle     Parse(string     s);
22 [C++]     public:     static     SqlSingle     Parse(String*     s);
23 [VB] Public Shared Function Parse(ByVal s As String) As SqlSingle
24 [JScript] public static function Parse(s : String) : SqlSingle;
25

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

[ . ] [ . ]

Subtract

```
[C#] public static SqlSingle Subtract(SqlSingle x, SqlSingle y);
[C++] public: static SqlSingle Subtract(SqlSingle x, SqlSingle y);
[VB] Public Shared Function Subtract(ByVal x As SqlSingle, ByVal y As
SqlSingle) As SqlSingle
[JavaScript] public static function Subtract(x : SqlSingle, y : SqlSingle) : SqlSingle;
```

*Description*

[ . ]

ToSqlBoolean

```
[C#] public SqlBoolean ToSqlBoolean();
[C++] public: SqlBoolean ToSqlBoolean();
[VB] Public Function ToSqlBoolean() As SqlBoolean
[JavaScript] public function ToSqlBoolean() : SqlBoolean;
```

*Description*

[ . ]

ToSqlByte

```
[C#] public SqlByte ToSqlByte();
```



1						
2	[C#]	public	SqlInt16	ToSqlInt16());		
3	[C++]	public:	SqlInt16	ToSqlInt16());		
4	[VB]	Public	Function	ToSqlInt16()	As	SqlInt16
5	[JScript]	public	function	ToSqlInt16()	:	SqlInt16;

6

7 *Description*

8 [ .]

9 ToSqlInt32

11	[C#]	public	SqlInt32	ToSqlInt32());		
12	[C++]	public:	SqlInt32	ToSqlInt32());		
13	[VB]	Public	Function	ToSqlInt32()	As	SqlInt32
14	[JScript]	public	function	ToSqlInt32()	:	SqlInt32;

15

16 *Description*

17 [ .]

18 ToSqlInt64

20	[C#]	public	SqlInt64	ToSqlInt64());		
21	[C++]	public:	SqlInt64	ToSqlInt64());		
22	[VB]	Public	Function	ToSqlInt64()	As	SqlInt64
23	[JScript]	public	function	ToSqlInt64()	:	SqlInt64;

24

25 *Description*

```

1      [ .]
2      ToSqlMoney
3
4      [C#]          public          SqlMoney          ToSqlMoney();
5      [C++]          public:          SqlMoney          ToSqlMoney();
6      [VB]      Public      Function      ToSqlMoney()      As      SqlMoney
7      [JScript]      public      function      ToSqlMoney()      :      SqlMoney;
8
9      Description
10     [ .]
11     ToSqlString
12
13     [C#]          public          SqlString          ToSqlString();
14     [C++]          public:          SqlString          ToSqlString();
15     [VB]      Public      Function      ToSqlString()      As      SqlString
16     [JScript]      public      function      ToSqlString()      :      SqlString;

```

```

17
18     Description
19     [ .]
20     ToString
21
22     [C#]          public          override          string          ToString();
23     [C++]          public:          String*          ToString();
24     [VB]      Overrides      Public      Function      ToString()      As      String
25     [JScript]      public      override      function      ToString()      :      String;

```

*Description*

[ . ] [ . ]  
 SqlString structure (System.Data.SqlTypes)  
 ToString

*Description*

Represents a variable-length stream of characters to be stored in or retrieved from the database.

ToString

[C#]	public	static	readonly	int	BinarySort;
[C++]	public:	static	int	BinarySort;	
[VB]	Public	Shared	ReadOnly	BinarySort	As Integer
[JScript]	public	static	var	BinarySort	: int;

*Description*

Specifies that sorts should be based on a characters numeric value rather than its alphabetic value.

**System.Data.SqlTypes.SqlString.BinarySort** functions as a constant for the **System.Data.SqlTypes.SqlString** class.

ToString

[C#]	public	static	readonly	int	IgnoreCase;
------	--------	--------	----------	-----	-------------

```

1  [C++]          public:          static          int          IgnoreCase;
2  [VB]   Public   Shared   ReadOnly   IgnoreCase   As   Integer
3  [JScript]   public   static   var   IgnoreCase   :   int;

```

#### 5 *Description*

6 Specifies that **SqlString** comparisons should ignore case.

7 **System.Data.SqlTypes.SqlString.IgnoreCase** functions as a constant for  
8 the **System.Data.SqlTypes.SqlString** class.

9 **ToString**

```

11 [C#]          public          static          readonly          int          IgnoreKanaType;
12 [C++]          public:          static          int          IgnoreKanaType;
13 [VB]   Public   Shared   ReadOnly   IgnoreKanaType   As   Integer
14 [JScript]   public   static   var   IgnoreKanaType   :   int;

```

#### 16 *Description*

17 Specifies that the string comparison must ignore the Kana type. Kana type  
18 refers to Japanese hiragana and katakana characters, which represent phonetic  
19 sounds in the Japanese language. Hiragana is used for native Japanese expressions  
20 and words, while katakana is used for words borrowed from other languages, such  
21 as "computer" or "internet". A phonetic sound can be expressed in both hiragana  
22 and katakana. If this value is selected, the hiragana character for one sound is  
23 considered equal to the katakana character for the same sound.

24 **System.Data.SqlTypes.SqlString.IgnoreKanaType** functions as a  
25 constant for the **System.Data.SqlTypes.SqlString** class.

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

Specifies that the string comparison must ignore the character width. For example, Japanese katakana characters can be written as full-width or half-width and, if this value is selected, the katakana characters written as full-width are considered equal to the same characters written in half-width.

**System.Data.SqlTypes.SqlString.IgnoreWidth** functions as a constant for the **System.Data.SqlTypes.SqlString** class.

ToString

[C#]	public	static	readonly	SqlString	Null;
[C++]	public:	static		SqlString	Null;
[VB]	Public	Shared	ReadOnly	Null	As SqlString
[JScript]	public	static	var	Null	: SqlString;

*Description*

Represents a null value that can be assigned to the **System.Data.SqlTypes.SqlString.Value** property of an instance of the **System.Data.SqlTypes.SqlString** structure.

**Null** functions as a constant for the **SqlString** structure.

SqlString

*Example Syntax:*

ToString

[C#]	public		SqlString(string	data);
[C++]	public:		SqlString(String*	data);
[VB]	Public	Sub	New(ByVal	data As String)

1 [JScript] public function SqlString(data : String);

2

3 *Description*

4 Initializes a new instance of the **System.Data.SqlTypes.SqlString**  
5 structure using the specified string. The string to store.

6 **SqlString**

7 *Example Syntax:*

8 **ToString**

9

10 [C#] public SqlString(string data, int lcid);

11 [C++] public: SqlString(String\* data, int lcid);

12 [VB] Public Sub New(ByVal data As String, ByVal lcid As Integer)

13 [JScript] public function SqlString(data : String, lcid : int);

14

15 *Description*

16 Initializes a new instance of the **System.Data.SqlTypes.SqlString**  
17 structure using the specified string and locale id values. The string to store.

18 Specifies the geographical locale and language for the new **SqlString** structure.

19 **SqlString**

20 *Example Syntax:*

21 **ToString**

22

23 [C#] public SqlString(int lcid, SqlCompareOptions compareOptions, byte[] data);

24 [C++] public: SqlString(int lcid, SqlCompareOptions compareOptions, unsigned

25 char data \_\_gc[]);

```

1  [VB] Public Sub New(ByVal lcid As Integer, ByVal compareOptions As
2  SqlCompareOptions,          ByVal          data()          As          Byte)
3  [JScript] public function SqlString(lcid : int, compareOptions :
4  SqlCompareOptions,          data          :          Byte[]);

```

#### 6 *Description*

7        Initializes a new instance of the **System.Data.SqlTypes.SqlString**  
8 structure using the specified locale id, compare options, and data. Specifies the  
9 geographical locale and language for the new **SqlString** structure. Specifies the  
10 compare options for the new **SqlString** structure. The data array to store.

11        **SqlString**

12        *Example Syntax:*

13        **ToString**

15 [C#] public SqlString(string data, int lcid, SqlCompareOptions compareOptions);

16 [C++] public: SqlString(String\* data, int lcid, SqlCompareOptions  
17 compareOptions);

18 [VB] Public Sub New(ByVal data As String, ByVal lcid As Integer, ByVal  
19 compareOptions As SqlCompareOptions)

20 [JScript] public function SqlString(data : String, lcid : int, compareOptions :  
21 SqlCompareOptions);

#### 23 *Description*

24        Initializes a new instance of the **System.Data.SqlTypes.SqlString**  
25 structure using the specified string, locale id, and compare option values. The

string to store. Specifies the geographical locale and language for the new **SqlString** structure. Specifies the compare options for the new **SqlString** structure.

**SqlString**

*Example Syntax:*

**ToString**

[C#] public **SqlString**(int **lcid**, **SqlCompareOptions** **compareOptions**, byte[] **data**,  
bool **fUnicode**);

[C++] public: **SqlString**(int **lcid**, **SqlCompareOptions** **compareOptions**, unsigned  
char **data** \_\_gc[], bool **fUnicode**);

[VB] Public Sub **New**(ByVal **lcid** As Integer, ByVal **compareOptions** As  
**SqlCompareOptions**, ByVal **data**() As Byte, ByVal **fUnicode** As Boolean)

[JScript] public function **SqlString**(**lcid** : int, **compareOptions** :  
**SqlCompareOptions**, **data** : Byte[], **fUnicode** : Boolean);

### *Description*

Initializes a new instance of the **System.Data.SqlTypes.SqlString** class.  
Specifies the geographical locale and language for the new **SqlString** structure.  
Specifies the compare options for the new **SqlString** structure. The data array to  
store. **true** if Unicode encoded, otherwise **false**.

**SqlString**

*Example Syntax:*

**ToString**



```

1 char data __gc[], int index, int count, bool fUnicode);
2 [VB] Public Sub New(ByVal lcId As Integer, ByVal compareOptions As
3 SqlCompareOptions, ByVal data() As Byte, ByVal index As Integer, ByVal count
4 As Integer, ByVal fUnicode As Boolean)
5 [JScript] public function SqlString(lcId : int, compareOptions :
6 SqlCompareOptions, data : Byte[], index : int, count : int, fUnicode : Boolean);

```

8 *Description*

9        Initializes a new instance of the **System.Data.SqlTypes.SqlString** class.

10       Specifies the geographical locale and language for the new **SqlString** structure.

11       Specifies the compare options for the new **SqlString** structure. The data array to

12       store. The starting index within the array. The number of characters from index to

13       copy. **true** if Unicode encoded, otherwise **false**.

14       CompareInfo

15       ToString

```

17 [C#]        public        CompareInfo        CompareInfo        {get;}
18 [C++]       public:       __property       CompareInfo*       get_CompareInfo();
19 [VB]        Public        ReadOnly        Property        CompareInfo        As        CompareInfo
20 [JScript]   public        function        get        CompareInfo()        :        CompareInfo;

```

22 *Description*

23       [ . ][ . ]

24       CultureInfo

25       ToString

```

1
2 [C#]          public          CultureInfo          CultureInfo          {get;}
3 [C++]         public:         __property          CultureInfo*        get_CultureInfo();
4 [VB]   Public  ReadOnly  Property  CultureInfo  As  CultureInfo
5 [JScript]  public  function  get  CultureInfo()  :  CultureInfo;

```

6  
7 *Description*

```

8     [ . ][ . ]
9     IsNull
10    ToString

```

```

12 [C#]          public          bool          IsNull          {get;}
13 [C++]         public:         __property          bool          get_IsNull();
14 [VB]   Public  ReadOnly  Property  IsNull  As  Boolean
15 [JScript]  public  function  get  IsNull()  :  Boolean;

```

16  
17 *Description*

```

18     Indicates whether the System.Data.SqlTypes.SqlString.Value of the
19 System.Data.SqlTypes.SqlString is System.Data.SqlTypes.SqlString.Null .
20     LCID
21     ToString

```

```

23 [C#]          public          int          LCID          {get;}
24 [C++]         public:         __property          int          get_LCID();
25 [VB]   Public  ReadOnly  Property  LCID  As  Integer

```

1 [JScript] public function get LCID() : int;

2

3 *Description*

4 Specifies the geographical locale and language for the **SqlString** structure.

5 SqlCompareOptions

6 ToString

7

8 [C#] public SqlCompareOptions SqlCompareOptions {get;}

9 [C++] public: \_\_property SqlCompareOptions get\_SqlCompareOptions();

10 [VB] Public ReadOnly Property SqlCompareOptions As SqlCompareOptions

11 [JScript] public function get SqlCompareOptions() : SqlCompareOptions;

12

13 *Description*

14 [ . ][ . ]

15 Value

16 ToString

17

18 [C#] public string Value {get;}

19 [C++] public: \_\_property String\* get\_Value();

20 [VB] Public ReadOnly Property Value As String

21 [JScript] public function get Value() : String;

22

23 *Description*

24 Gets the string that is stored in this **System.Data.SqlTypes.SqlString**

25 structure. This property is read-only.

```

1      Clone
2
3  [C#]          public          SqlString          Clone();
4  [C++]         public:         SqlString          Clone();
5  [VB]         Public          Function          Clone()      As      SqlString
6  [JScript]     public          function          Clone()      :      SqlString;
7

```

#### 8 *Description*

9       Creates a copy of this **System.Data.SqlTypes.SqlString** object.  
10 *Return Value:* A new **System.Data.SqlTypes.SqlString** object in which all  
11 property values are the same as the original.

12       CompareOptionsFromSqlCompareOptions

```

13
14 [C#]          public          static          CompareOptions
15 CompareOptionsFromSqlCompareOptions(SqlCompareOptions compareOptions);
16 [C++]         public:         static          CompareOptions
17 CompareOptionsFromSqlCompareOptions(SqlCompareOptions compareOptions);
18 [VB] Public Shared Function CompareOptionsFromSqlCompareOptions(ByVal
19 compareOptions As SqlCompareOptions) As CompareOptions
20 [JScript]     public          static          function
21 CompareOptionsFromSqlCompareOptions(compareOptions          :
22 SqlCompareOptions)          :          CompareOptions;
23

```

#### 24 *Description*

25       [ .]

## CompareTo

```
[C#]      public      int      CompareTo(object      value);
[C++]     public:     __sealed  int      CompareTo(Object*      value);
[VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
Integer
[JScript] public  function  CompareTo(value  :  Object)  :  int;
```

### *Description*

Compares this instance of **System.Data.SqlTypes.SqlString** to the supplied object and returns an indication of their relative values.

*Return Value:* A signed number indicating the relative values of the instance and the object. The object to be compared.

## Concat

```
[C#]      public      static  SqlString  Concat(SqlString  x,  SqlString  y);
[C++]     public:     static  SqlString  Concat(SqlString  x,  SqlString  y);
[VB] Public Shared Function Concat(ByVal x As SqlString, ByVal y As
SqlString)
As
SqlString
[JScript] public static function Concat(x : SqlString, y : SqlString) : SqlString;
```

### *Description*

[ .]

## Equals

```

1
2 [C#]      public      override      bool      Equals(object      value);
3 [C++]      public:      bool      Equals(Object*      value);
4 [VB] Overrides Public Function Equals(ByVal value As Object) As Boolean
5 [JScript] public override function Equals(value : Object) : Boolean;
6

```

7 *Description*

8 Compares the supplied object parameter to the  
9 **System.Data.SqlTypes.SqlString.Value** property of the  
10 **System.Data.SqlTypes.SqlString** object.

11 *Return Value:* Equals will return **true** if the object is an instance of  
12 **System.Data.SqlTypes.SqlString** and the two are equal; otherwise **false** . The  
13 object to be compared.

14 Equals

```

15
16 [C#] public static new SqlBoolean Equals(SqlString x, SqlString y);
17 [C++] public: static SqlBoolean Equals(SqlString x, SqlString y);
18 [VB] Shadows Public Shared Function Equals(ByVal x As SqlString, ByVal y As
19 SqlString) As SqlBoolean
20 [JScript] public static hide function Equals(x : SqlString, y : SqlString) :
21 SqlBoolean;
22

```

23 *Description*

24 [ .]  
25 GetHashCode

```

1
2 [C#]          public          override          int          GetHashCode();
3 [C++]          public:          int          GetHashCode();
4 [VB]  Overrides  Public  Function  GetHashCode()  As  Integer
5 [JScript]  public  override  function  GetHashCode()  :  int;

```

6  
7 *Description*

8 Gets the hash code for this instance.

9 *Return Value:* A 32-bit signed integer hash code.

10 **GetNonUnicodeBytes**

```

11
12 [C#]          public          byte[]          GetNonUnicodeBytes();
13 [C++]  public:  unsigned  char  GetNonUnicodeBytes()  __gc[];
14 [VB]  Public  Function  GetNonUnicodeBytes()  As  Byte()
15 [JScript]  public  function  GetNonUnicodeBytes()  :  Byte[];

```

16  
17 *Description*

18 Returns an array of bytes, containing the contents of the

19 **System.Data.SqlTypes.SqlString** in ANSI format.

20 *Return Value:* An byte array, containing the contents of the

21 **System.Data.SqlTypes.SqlString** in ANSI format.

22 **GetUnicodeBytes**

```

23
24 [C#]          public          byte[]          GetUnicodeBytes();
25 [C++]  public:  unsigned  char  GetUnicodeBytes()  __gc[];

```

```

1  [VB]      Public      Function      GetUnicodeBytes()      As      Byte()
2  [JScript]      public      function      GetUnicodeBytes()      :      Byte[];

```

3

4 *Description*

5 Returns an array of bytes, containing the contents of the

6 **System.Data.SqlTypes.SqlString** in Unicode format.

7 *Return Value:* An byte array, containing the contents of the

8 **System.Data.SqlTypes.SqlString** in Unicode format.

9 GreaterThan

10

```

11 [C#] public static SqlBoolean GreaterThan(SqlString x, SqlString y);
12 [C++] public: static SqlBoolean GreaterThan(SqlString x, SqlString y);
13 [VB] Public Shared Function GreaterThan(ByVal x As SqlString, ByVal y As
14 SqlString) As SqlBoolean
15 [JScript] public static function GreaterThan(x : SqlString, y : SqlString) :
16 SqlBoolean;

```

17

18 *Description*

19 [ .]

20 GreaterThanOrEqual

21

```

22 [C#] public static SqlBoolean GreaterThanOrEqual(SqlString x, SqlString y);
23 [C++] public: static SqlBoolean GreaterThanOrEqual(SqlString x, SqlString y);
24 [VB] Public Shared Function GreaterThanOrEqual(ByVal x As SqlString, ByVal
25 y As SqlString) As SqlBoolean

```

1 [JScript] public static function GreaterThanOrEqual(x : SqlString, y : SqlString) :  
2 SqlBoolean;

3  
4 *Description*

5 [ .]  
6 LessThan

7  
8 [C#] public static SqlBoolean LessThan(SqlString x, SqlString y);

9 [C++] public: static SqlBoolean LessThan(SqlString x, SqlString y);

10 [VB] Public Shared Function LessThan(ByVal x As SqlString, ByVal y As  
11 SqlString) As SqlBoolean

12 [JScript] public static function LessThan(x : SqlString, y : SqlString) :  
13 SqlBoolean;

14  
15 *Description*

16 [ .]  
17 LessThanOrEqual

18  
19 [C#] public static SqlBoolean LessThanOrEqual(SqlString x, SqlString y);

20 [C++] public: static SqlBoolean LessThanOrEqual(SqlString x, SqlString y);

21 [VB] Public Shared Function LessThanOrEqual(ByVal x As SqlString, ByVal y  
22 As SqlString) As SqlBoolean

23 [JScript] public static function LessThanOrEqual(x : SqlString, y : SqlString) :  
24 SqlBoolean;

25

*Description*

[ .]

NotEquals

[C#] public static SqlBoolean NotEquals(SqlString x, SqlString y);

[C++] public: static SqlBoolean NotEquals(SqlString x, SqlString y);

[VB] Public Shared Function NotEquals(ByVal x As SqlString, ByVal y As  
SqlString) As SqlBoolean

[JScript] public static function NotEquals(x : SqlString, y : SqlString) :  
SqlBoolean;

*Description*

[ .]

op\_Addition

[C#] public static SqlString operator +(SqlString x, SqlString y);

[C++] public: static SqlString op\_Addition(SqlString x, SqlString y);

[VB] returnValue = SqlString.op\_Addition(x, y)

[JScript] returnValue = x + y;

*Description*

Concatenates the two **System.Data.SqlTypes.SqlString** operands.

*Return Value:* A **System.Data.SqlTypes.SqlString** containing the newly  
concatenated value representing the contents of the two

1 **System.Data.SqlTypes.SqlString** parameters. A

2 **System.Data.SqlTypes.SqlString**. A **System.Data.SqlTypes.SqlString**.

3 **op\_Equality**

5 [C#] public static SqlBoolean operator ==(SqlString x, SqlString y);

6 [C++] public: static SqlBoolean op\_Equality(SqlString x, SqlString y);

7 [VB] returnValue = SqlString.op\_Equality(x, y)

8 [JScript] returnValue = x == y;

10 *Description*

11 Performs a logical comparison of the two  
12 **System.Data.SqlTypes.SqlString** operands to determine if they are equal.

13 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is

14 **System.Data.SqlTypes.SqlBoolean.True** if the two instances are equal or

15 **System.Data.SqlTypes.SqlBoolean.False** if the two instances are not equal. If

16 either instance of **System.Data.SqlTypes.SqlString** is null, the

17 **System.Data.SqlTypes.SqlBoolean.Value** of the

18 **System.Data.SqlTypes.SqlBoolean** will be

19 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlString**.

20 A **System.Data.SqlTypes.SqlString**.

21 **op\_Explicit**

23 [C#] public static explicit operator SqlString(SqlBoolean x);

24 [C++] public: static SqlString op\_Explicit(SqlBoolean x);

25 [VB] returnValue = SqlString.op\_Explicit(x)

1 [JScript]                   returnValue                   =                   SqlString(x);

2

3 *Description*

4       Converts       the       **System.Data.SqlTypes.SqlBit**       parameter       to  
5 **System.Data.SqlTypes.SqlString** .

6 *Return Value:* A new **System.Data.SqlTypes.SqlString** containing the string  
7 representation of the **System.Data.SqlTypes.SqlBit** parameter. The  
8 **System.Data.SqlTypes.SqlBit** structure to be converted.

9       op\_Explicit

10

11 [C#]       public       static       explicit       operator       SqlString(SqlByte       x);

12 [C++]       public:       static       SqlString       op\_Explicit(SqlByte       x);

13 [VB]               returnValue               =               SqlString.op\_Explicit(x)

14 [JScript]               returnValue               =               SqlString(x);

15

16 *Description*

17       Converts the supplied **System.Data.SqlTypes.SqlByte** structure to  
18 **System.Data.SqlTypes.SqlString** .

19 *Return Value:* A new **System.Data.SqlTypes.SqlString** object containing the  
20 string representation of the **System.Data.SqlTypes.SqlByte** parameter. The  
21 **System.Data.SqlTypes.SqlByte** structure to be converted.

22       op\_Explicit

23

24 [C#]       public       static       explicit       operator       SqlString(SqlDateTime       x);

25 [C++]       public:       static       SqlString       op\_Explicit(SqlDateTime       x);

```

1  [VB]          returnValue          =          SqlString.op_Explicit(x)
2  [JScript]     returnValue          =          SqlString(x);

```

3

4 *Description*

5       Converts the supplied **System.Data.SqlTypes.SqlDateTime** parameter to

6 **System.Data.SqlTypes.SqlString**

7 *Return Value:* A new **System.Data.SqlTypes.SqlString** containing the string

8 representation of the **System.Data.SqlTypes.SqlDateTime** parameter. The

9 **System.Data.SqlTypes.SqlDateTime** structure to be converted.

10       op\_Explicit

```

11
12 [C#]   public   static   explicit   operator   SqlString(SqlDecimal   x);
13 [C++]  public:   static   SqlString   op_Explicit(SqlDecimal   x);
14 [VB]   returnValue          =          SqlString.op_Explicit(x)
15 [JScript]     returnValue          =          SqlString(x);

```

16

17 *Description*

18       Converts the supplied **System.Data.SqlTypes.SqlDecimal** parameter to

19 **System.Data.SqlTypes.SqlString**

20 *Return Value:* A new **System.Data.SqlTypes.SqlString** containing the string

21 representation of the **System.Data.SqlTypes.SqlDecimal** parameter.

22       op\_Explicit

```

23
24 [C#]   public   static   explicit   operator   SqlString(SqlDouble   x);
25 [C++]  public:   static   SqlString   op_Explicit(SqlDouble   x);

```

```

1  [VB]          returnValue          =          SqlString.op_Explicit(x)
2  [JScript]          returnValue          =          SqlString(x);

```

3

4 *Description*

5       Converts the supplied **System.Data.SqlTypes.SqlDouble** parameter to

6 **System.Data.SqlTypes.SqlString**

7 *Return Value:* A new **System.Data.SqlTypes.SqlString** containing the string

8 representation of the **System.Data.SqlTypes.SqlDouble** parameter. The

9 **System.Data.SqlTypes.SqlDouble** structure to be converted.

10       op\_Explicit

```

11
12 [C#]   public   static   explicit   operator   SqlString(SqlGuid   x);
13 [C++]   public:   static   SqlString   op_Explicit(SqlGuid   x);
14 [VB]          returnValue          =          SqlString.op_Explicit(x)
15 [JScript]          returnValue          =          SqlString(x);

```

16

17 *Description*

18       Converts the supplied **System.Data.SqlTypes.SqlGuid** parameter to

19 **System.Data.SqlTypes.SqlString** . The **System.Data.SqlTypes.SqlGuid**

20 structure to be converted.

21       op\_Explicit

```

22
23 [C#]   public   static   explicit   operator   SqlString(SqlInt16   x);
24 [C++]   public:   static   SqlString   op_Explicit(SqlInt16   x);
25 [VB]          returnValue          =          SqlString.op_Explicit(x)

```

1 [JScript]                      returnValue                      =                      SqlString(x);

2

3 *Description*

4        Converts the supplied **System.Data.SqlTypes.SqlInt16** parameter to  
5 **System.Data.SqlTypes.SqlString**

6 *Return Value:* A new **System.Data.SqlTypes.SqlString** object containing the  
7 string representation of the **System.Data.SqlTypes.SqlInt16** parameter. The  
8 **System.Data.SqlTypes.SqlInt16** structure to be converted.

9        op\_Explicit

10

11 [C#]        public        static        explicit        operator        SqlString(SqlInt32        x);

12 [C++]        public:        static        SqlString        op\_Explicit(SqlInt32        x);

13 [VB]                      returnValue                      =                      SqlString.op\_Explicit(x)

14 [JScript]                      returnValue                      =                      SqlString(x);

15

16 *Description*

17        Converts the supplied **System.Data.SqlTypes.SqlInt32** parameter to  
18 **System.Data.SqlTypes.SqlString**

19 *Return Value:* A new **System.Data.SqlTypes.SqlString** object containing the  
20 string representation of the **System.Data.SqlTypes.SqlInt32** parameter. The  
21 **SqlInt32** structure to be converted.

22        op\_Explicit

23

24 [C#]        public        static        explicit        operator        SqlString(SqlInt64        x);

25 [C++]        public:        static        SqlString        op\_Explicit(SqlInt64        x);

```

1  [VB]          returnValue          =          SqlString.op_Explicit(x)
2  [JScript]     returnValue          =          SqlString(x);

```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlInt64** parameter to **System.Data.SqlTypes.SqlString**.

*Return Value:* A new **System.Data.SqlTypes.SqlString** object containing the string representation of the **System.Data.SqlTypes.SqlInt64** parameter. The **System.Data.SqlTypes.SqlInt64** structure to be converted.

op\_Explicit

```

12 [C#]   public   static   explicit   operator   SqlString(SqlMoney   x);
13 [C++]  public:   static   SqlString   op_Explicit(SqlMoney   x);
14 [VB]   returnValue          =          SqlString.op_Explicit(x)
15 [JScript]     returnValue          =          SqlString(x);

```

#### *Description*

Converts the supplied **System.Data.SqlTypes.SqlMoney** parameter to **System.Data.SqlTypes.SqlString**.

*Return Value:* A new **System.Data.SqlTypes.SqlString** containing the string representation of the **System.Data.SqlTypes.SqlMoney** parameter. The **System.Data.SqlTypes.SqlMoney** structure to be converted.

op\_Explicit

```

25 [C#]   public   static   explicit   operator   SqlString(SqlSingle   x);

```

```

1  [C++]      public:      static      SqlString      op_Explicit(SqlSingle      x);
2  [VB]              returnValue      =              SqlString.op_Explicit(x)
3  [JScript]              returnValue      =              SqlString(x);

```

#### 5 *Description*

6 Converts the supplied **System.Data.SqlTypes.SqlSingle** parameter to  
7 **System.Data.SqlTypes.SqlString**

8 *Return Value:* A new **System.Data.SqlTypes.SqlString** containing the string  
9 representation of the **System.Data.SqlTypes.SqlSingle** parameter. The  
10 **System.Data.SqlTypes.SqlSingle** structure to be converted.

11 op\_Explicit

```

12
13 [C#]      public      static      explicit      operator      string(SqlString      x);
14 [C++]      public:              static              String*              op_Explicit();
15 [VB]              returnValue      =              SqlString.op_Explicit(x)
16 [JScript]              returnValue      =              String(x);

```

#### 18 *Description*

19 Converts a **System.Data.SqlTypes.SqlString** to a **System.String**

20 *Return Value:* A **System.String** , whose contents are the same as the  
21 **System.Data.SqlTypes.SqlString.Value** property of the  
22 **System.Data.SqlTypes.SqlString** parameter. The  
23 **System.Data.SqlTypes.SqlString** to be converted.

24 op\_GreaterThan

25

```

1
2 [C#] public static SqlBoolean operator >(SqlString x, SqlString y);
3 [C++] public: static SqlBoolean op_GreaterThan(SqlString x, SqlString y);
4 [VB]     returnValue      =      SqlString.op_GreaterThan(x,      y)
5 [JScript]     returnValue      =      x      >      y;

```

### 7 *Description*

8 Performs a logical comparison of the two  
9 **System.Data.SqlTypes.SqlString** operands to determine if the first is greater than  
10 the second.

11 *Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is  
12 **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than the  
13 second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either  
14 instance of **System.Data.SqlTypes.SqlString** is null, the  
15 **System.Data.SqlTypes.SqlBoolean.Value** of the  
16 **System.Data.SqlTypes.SqlBoolean** will be  
17 **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlString**.  
18 A **System.Data.SqlTypes.SqlString**.

19 op\_GreaterThanOrEqual

```

20
21 [C#] public static SqlBoolean operator >=(SqlString x, SqlString y);
22 [C++] public: static SqlBoolean op_GreaterThanOrEqual(SqlString x, SqlString
23 y);
24 [VB]     returnValue      =      SqlString.op_GreaterThanOrEqual(x,      y)
25 [JScript]     returnValue      =      x      >=      y;

```

## Description

Performs a logical comparison of the two **System.Data.SqlTypes.SqlString** operands to determine if the first is greater than or equal to the second.

**Return Value:** A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is greater than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False**. If either instance of **System.Data.SqlTypes.SqlString** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlString**. A **System.Data.SqlTypes.SqlString**.

op\_Implicit

[C#] public static implicit operator SqlString(string x);

[C++] public: static SqlString op\_Implicit(String\* x);

[VB] returnValue = SqlString.op\_Implicit(x)

[JScript] returnValue = x;

## Description

Converts the **System.String** parameter to a **System.Data.SqlTypes.SqlString**. The **System.String** to be converted.

op\_Inequality

```

1
2 [C#] public static SqlBoolean operator !=(SqlString x, SqlString y);
3 [C++] public: static SqlBoolean op_Inequality(SqlString x, SqlString y);
4 [VB]     returnValue      =      SqlString.op_Inequality(x,      y)
5 [JScript]     returnValue      =      x      !=      y;
6

```

### *Description*

Performs a logical comparison of the two **System.Data.SqlTypes.SqlString** operands to determine if they are equal.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the two instances are not equal or **System.Data.SqlTypes.SqlBoolean.False** if the two instances are equal. If either instance of **System.Data.SqlTypes.SqlString** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null**. A **System.Data.SqlTypes.SqlString**.

A **System.Data.SqlTypes.SqlString**.

op\_LessThan

```

19
20 [C#]     public     static     SqlBoolean     operator
21 [C++] public: static SqlBoolean op_LessThan(SqlString x, SqlString y);
22 [VB]     returnValue      =      SqlString.op_LessThan(x,      y)
23 [JScript]     returnValue      =      x      <      y;
24

```

### *Description*

Performs a logical comparison of the two **System.Data.SqlTypes.SqlString** operands to determine if the first is less than the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If either instance of **System.Data.SqlTypes.SqlString** is null, the **System.Data.SqlTypes.SqlBoolean.Value** of the **System.Data.SqlTypes.SqlBoolean** will be **System.Data.SqlTypes.SqlBoolean.Null** . A **System.Data.SqlTypes.SqlString**.  
A **System.Data.SqlTypes.SqlString**.

op\_LessThanOrEqual

[C#] public static SqlBoolean operator <=(SqlString x, SqlString y);

[C++] public: static SqlBoolean op\_LessThanOrEqual(SqlString x, SqlString y);

[VB] returnValue = SqlString.op\_LessThanOrEqual(x, y)

[JScript] returnValue = x <= y;

### Description

Performs a logical comparison of the two **System.Data.SqlTypes.SqlString** operands to determine if the first is less than or equal to the second.

*Return Value:* A **System.Data.SqlTypes.SqlBoolean** that is **System.Data.SqlTypes.SqlBoolean.True** if the first instance is less than or equal to the second instance, otherwise **System.Data.SqlTypes.SqlBoolean.False** . If



```

1
2 [C#]          public          SqlDateTime          ToSqlDateTime();
3 [C++]         public:         SqlDateTime          ToSqlDateTime();
4 [VB]   Public   Function      ToSqlDateTime()      As      SqlDateTime
5 [JScript]   public   function  ToSqlDateTime()      :      SqlDateTime;

```

6  
7 *Description*

```

8     [ .]
9     ToSqlDecimal

```

```

11 [C#]          public          SqlDecimal           ToSqlDecimal();
12 [C++]         public:         SqlDecimal           ToSqlDecimal();
13 [VB]   Public   Function      ToSqlDecimal()      As      SqlDecimal
14 [JScript]   public   function  ToSqlDecimal()      :      SqlDecimal;

```

15  
16 *Description*

```

17     [ .]
18     ToSqlDouble

```

```

20 [C#]          public          SqlDouble            ToSqlDouble();
21 [C++]         public:         SqlDouble            ToSqlDouble();
22 [VB]   Public   Function      ToSqlDouble()      As      SqlDouble
23 [JScript]   public   function  ToSqlDouble()      :      SqlDouble;

```

24  
25 *Description*

1	[ . ]					
2	ToSqlGuid					
3						
4	[C#]	public		SqlGuid		ToSqlGuid();
5	[C++]	public:		SqlGuid		ToSqlGuid();
6	[VB]	Public	Function	ToSqlGuid()	As	SqlGuid
7	[JScript]	public	function	ToSqlGuid()	:	SqlGuid;

*Description*

10	[ .]					
11	ToSqlInt16					
12						
13	[C#]	public		SqlInt16		ToSqlInt16();
14	[C++]	public:		SqlInt16		ToSqlInt16();
15	[VB]	Public	Function	ToSqlInt16()	As	SqlInt16
16	[JScript]	public	function	ToSqlInt16()	:	SqlInt16;

*Description*

19	[ .]					
20	ToSqlInt32					
21						
22	[C#]	public		SqlInt32		ToSqlInt32();
23	[C++]	public:		SqlInt32		ToSqlInt32();
24	[VB]	Public	Function	ToSqlInt32()	As	SqlInt32
25	[JScript]	public	function	ToSqlInt32()	:	SqlInt32;

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

[ .]

ToSqlInt64

[C#]	public	SqlInt64	ToSqlInt64();
[C++]	public:	SqlInt64	ToSqlInt64();
[VB]	Public	Function	ToSqlInt64() As SqlInt64
[JScript]	public	function	ToSqlInt64() : SqlInt64;

*Description*

[ .]

ToSqlMoney

[C#]	public	SqlMoney	ToSqlMoney();
[C++]	public:	SqlMoney	ToSqlMoney();
[VB]	Public	Function	ToSqlMoney() As SqlMoney
[JScript]	public	function	ToSqlMoney() : SqlMoney;

*Description*

[ .]

ToSqlSingle

[C#]	public	SqlSingle	ToSqlSingle();
[C++]	public:	SqlSingle	ToSqlSingle();

1 [VB] Public Function ToSqlSingle() As SqlSingle  
 2 [JScript] public function ToSqlSingle() : SqlSingle;

3  
 4 *Description*

5 [ .]  
 6 ToString

8 [C#] public override string ToString();

9 [C++] public: String\* ToString();

10 [VB] Overrides Public Function ToString() As String

11 [JScript] public override function ToString() : String; Converts a

12 **System.Data.SqlTypes.SqlString** object to a **System.String**.

13  
 14 *Description*

15 Converts a **System.Data.SqlTypes.SqlString** object to a **System.String**.

16 **SqlTruncateException** class (System.Data.SqlTypes)

17 ToString

18  
 19  
 20 *Description*

21 The exception that is thrown when setting a value into a **SqlType** structure  
 22 would truncate that value.

23 **SqlTruncateException**

24 *Example Syntax:*

25 ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#]                public                SqlTruncateException();
[C++]                public:                SqlTruncateException();
[VB]                Public                Sub                New()
[JavaScript] public function SqlTruncateException(); Initializes a new instance of the
System.Data.SqlTypes.SqlTruncateException class.
```

*Description*

Initializes a new instance of the  
**System.Data.SqlTypes.SqlTruncateException** class with default properties.

SqlTruncateException

*Example Syntax:*

ToString

```
[C#]                public                SqlTruncateException(string                message);
[C++]                public:                SqlTruncateException(String*                message);
[VB]                Public                Sub                New(ByVal                message                As                String)
[JavaScript] public function SqlTruncateException(message : String);
```

*Description*

Initializes a new instance of the  
**System.Data.SqlTypes.SqlTruncateException** class with a specified error  
message. The error message that explains the reason for the exception.

HelpLink

HResult

1        InnerException  
 2        Message  
 3        Source  
 4        StackTrace  
 5        TargetSite  
 6        ISerializable.GetObjectData

8    [C#] void ISerializable.GetObjectData(SerializationInfo si, StreamingContext  
 9    context);

10   [C++] void ISerializable::GetObjectData(SerializationInfo\* si, StreamingContext  
 11   context);

12   [VB] Sub GetObjectData(ByVal si As SerializationInfo, ByVal context As  
 13   StreamingContext)                      Implements                      ISerializable.GetObjectData

14   [JScript] function ISerializable.GetObjectData(si : SerializationInfo, context :  
 15   StreamingContext);

16        SqlTypeException class (System.Data.SqlTypes)

17        ToString

20    *Description*

21        The base exception class for the **System.Data.SqlTypes** .

22        SqlTypeException

23        *Example Syntax:*

24        ToString

25

```

1
2 [C#]          public          SqlTypeException(string          message);
3 [C++]         public:         SqlTypeException(String*         message);
4 [VB]   Public   Sub   New(ByVal   message   As   String)
5 [JScript]   public   function   SqlTypeException(message   :   String);
6

```

#### 7 *Description*

8        Initializes        a        new        instance        of        the  
9 **System.Data.SqlTypes.SqlTypeException**

#### 12 EXEMPLARY COMPUTING SYSTEM AND ENVIRONMENT

13        Fig. 4 illustrates an example of a suitable computing environment 400  
14 within which the programming framework 132 may be implemented (either fully  
15 or partially). The computing environment 400 may be utilized in the computer  
16 and network architectures described herein.

17        The exemplary computing environment 400 is only one example of a  
18 computing environment and is not intended to suggest any limitation as to the  
19 scope of use or functionality of the computer and network architectures. Neither  
20 should the computing environment 400 be interpreted as having any dependency  
21 or requirement relating to any one or combination of components illustrated in the  
22 exemplary computing environment 400.

23        The framework 132 may be implemented with numerous other general  
24 purpose or special purpose computing system environments or configurations.  
25 Examples of well known computing systems, environments, and/or configurations

1 that may be suitable for use include, but are not limited to, personal computers,  
2 server computers, multiprocessor systems, microprocessor-based systems, network  
3 PCs, minicomputers, mainframe computers, distributed computing environments  
4 that include any of the above systems or devices, and so on. Compact or subset  
5 versions of the framework may also be implemented in clients of limited  
6 resources, such as cellular phones, personal digital assistants, handheld computers,  
7 or other communication/computing devices.

8 The framework 132 may be described in the general context of computer-  
9 executable instructions, such as program modules, being executed by one or more  
10 computers or other devices. Generally, program modules include routines,  
11 programs, objects, components, data structures, etc. that perform particular tasks  
12 or implement particular abstract data types. The framework 132 may also be  
13 practiced in distributed computing environments where tasks are performed by  
14 remote processing devices that are linked through a communications network. In  
15 a distributed computing environment, program modules may be located in both  
16 local and remote computer storage media including memory storage devices.

17 The computing environment 400 includes a general-purpose computing  
18 device in the form of a computer 402. The components of computer 402 can  
19 include, by are not limited to, one or more processors or processing units 404, a  
20 system memory 406, and a system bus 408 that couples various system  
21 components including the processor 404 to the system memory 406.

22 The system bus 408 represents one or more of several possible types of bus  
23 structures, including a memory bus or memory controller, a peripheral bus, an  
24 accelerated graphics port, and a processor or local bus using any of a variety of  
25 bus architectures. By way of example, such architectures can include an Industry

1 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an  
2 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)  
3 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a  
4 Mezzanine bus.

5 Computer 402 typically includes a variety of computer readable media.  
6 Such media can be any available media that is accessible by computer 402 and  
7 includes both volatile and non-volatile media, removable and non-removable  
8 media.

9 The system memory 406 includes computer readable media in the form of  
10 volatile memory, such as random access memory (RAM) 410, and/or non-volatile  
11 memory, such as read only memory (ROM) 412. A basic input/output system  
12 (BIOS) 414, containing the basic routines that help to transfer information  
13 between elements within computer 402, such as during start-up, is stored in ROM  
14 412. RAM 410 typically contains data and/or program modules that are  
15 immediately accessible to and/or presently operated on by the processing unit 404.

16 Computer 402 may also include other removable/non-removable,  
17 volatile/non-volatile computer storage media. By way of example, Fig. 4  
18 illustrates a hard disk drive 416 for reading from and writing to a non-removable,  
19 non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading  
20 from and writing to a removable, non-volatile magnetic disk 420 (e.g., a "floppy  
21 disk"), and an optical disk drive 422 for reading from and/or writing to a  
22 removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other  
23 optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk  
24 drive 422 are each connected to the system bus 408 by one or more data media  
25 interfaces 426. Alternatively, the hard disk drive 416, magnetic disk drive 418,

1 and optical disk drive 422 can be connected to the system bus 408 by one or more  
2 interfaces (not shown).

3 The disk drives and their associated computer-readable media provide non-  
4 volatile storage of computer readable instructions, data structures, program  
5 modules, and other data for computer 402. Although the example illustrates a  
6 hard disk 416, a removable magnetic disk 420, and a removable optical disk 424,  
7 it is to be appreciated that other types of computer readable media which can store  
8 data that is accessible by a computer, such as magnetic cassettes or other magnetic  
9 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or  
10 other optical storage, random access memories (RAM), read only memories  
11 (ROM), electrically erasable programmable read-only memory (EEPROM), and  
12 the like, can also be utilized to implement the exemplary computing system and  
13 environment.

14 Any number of program modules can be stored on the hard disk 416,  
15 magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by  
16 way of example, an operating system 426, one or more application programs 428,  
17 other program modules 430, and program data 432. Each of the operating system  
18 426, one or more application programs 428, other program modules 430, and  
19 program data 432 (or some combination thereof) may include elements of the  
20 programming framework 132.

21 A user can enter commands and information into computer 402 via input  
22 devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse").  
23 Other input devices 438 (not shown specifically) may include a microphone,  
24 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and  
25 other input devices are connected to the processing unit 404 via input/output

1 interfaces 440 that are coupled to the system bus 408, but may be connected by  
2 other interface and bus structures, such as a parallel port, game port, or a universal  
3 serial bus (USB).

4 A monitor 442 or other type of display device can also be connected to the  
5 system bus 408 via an interface, such as a video adapter 444. In addition to the  
6 monitor 442, other output peripheral devices can include components such as  
7 speakers (not shown) and a printer 446 which can be connected to computer 402  
8 via the input/output interfaces 440.

9 Computer 402 can operate in a networked environment using logical  
10 connections to one or more remote computers, such as a remote computing device  
11 448. By way of example, the remote computing device 448 can be a personal  
12 computer, portable computer, a server, a router, a network computer, a peer device  
13 or other common network node, and so on. The remote computing device 448 is  
14 illustrated as a portable computer that can include many or all of the elements and  
15 features described herein relative to computer 402.

16 Logical connections between computer 402 and the remote computer 448  
17 are depicted as a local area network (LAN) 450 and a general wide area network  
18 (WAN) 452. Such networking environments are commonplace in offices,  
19 enterprise-wide computer networks, intranets, and the Internet.

20 When implemented in a LAN networking environment, the computer 402 is  
21 connected to a local network 450 via a network interface or adapter 454. When  
22 implemented in a WAN networking environment, the computer 402 typically  
23 includes a modem 456 or other means for establishing communications over the  
24 wide network 452. The modem 456, which can be internal or external to computer  
25 402, can be connected to the system bus 408 via the input/output interfaces 440 or

1 other appropriate mechanisms. It is to be appreciated that the illustrated network  
2 connections are exemplary and that other means of establishing communication  
3 link(s) between the computers 402 and 448 can be employed.

4 In a networked environment, such as that illustrated with computing  
5 environment 400, program modules depicted relative to the computer 402, or  
6 portions thereof, may be stored in a remote memory storage device. By way of  
7 example, remote application programs 458 reside on a memory device of remote  
8 computer 448. For purposes of illustration, application programs and other  
9 executable program components such as the operating system are illustrated herein  
10 as discrete blocks, although it is recognized that such programs and components  
11 reside at various times in different storage components of the computing device  
12 402, and are executed by the data processor(s) of the computer.

13 An implementation of the framework 132, and particularly, the API 142 or  
14 calls made to the API 142, may be stored on or transmitted across some form of  
15 computer readable media. Computer readable media can be any available media  
16 that can be accessed by a computer. By way of example, and not limitation,  
17 computer readable media may comprise "computer storage media" and  
18 "communications media." "Computer storage media" include volatile and non-  
19 volatile, removable and non-removable media implemented in any method or  
20 technology for storage of information such as computer readable instructions, data  
21 structures, program modules, or other data. Computer storage media includes, but  
22 is not limited to, RAM, ROM, EEPROM, flash memory or other memory  
23 technology, CD-ROM, digital versatile disks (DVD) or other optical storage,  
24 magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage  
25

1 devices, or any other medium which can be used to store the desired information  
2 and which can be accessed by a computer.

3 "Communication media" typically embodies computer readable  
4 instructions, data structures, program modules, or other data in a modulated data  
5 signal, such as carrier wave or other transport mechanism. Communication media  
6 also includes any information delivery media. The term "modulated data signal"  
7 means a signal that has one or more of its characteristics set or changed in such a  
8 manner as to encode information in the signal. By way of example, and not  
9 limitation, communication media includes wired media such as a wired network or  
10 direct-wired connection, and wireless media such as acoustic, RF, infrared, and  
11 other wireless media. Combinations of any of the above are also included within  
12 the scope of computer readable media.

13 Alternatively, portions of the framework may be implemented in hardware  
14 or a combination of hardware, software, and/or firmware. For example, one or  
15 more application specific integrated circuits (ASICs) or programmable logic  
16 devices (PLDs) could be designed or programmed to implement one or more  
17 portions of the framework.

### 18 19 Conclusion

20 Although the invention has been described in language specific to structural  
21 features and/or methodological acts, it is to be understood that the invention  
22 defined in the appended claims is not necessarily limited to the specific features or  
23 acts described. Rather, the specific features and acts are disclosed as exemplary  
24 forms of implementing the claimed invention.